МОДЕЛИРОВАНИЕ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Федеральное агентство по образованию

Уральский государственный технический университет – УПИ
имени первого Президента России Б.Н. Ельцина

В.Г. Лисиенко, Н.Г. Дружинина, О.Г. Трофимова, С.П. Трофимов

МОДЕЛИРОВАНИЕ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Учебное пособие

Научный редактор доц., канд. техн. наук В.А. Морозова

Рекомендовано Региональным отделением УрФО учебно-методического объединения вузов Российской Федерации по образованию в области радиотехники, электроники, биомедицинской техники и автоматизации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 220200 – Автоматизация и управление в УрФО

Екатеринбург

УГТУ-УПИ

2009

УДК 004.94(075.8)
ББК 32.973.26-018я73
М74

Рецензенты:
кафедра информационных технологий Института урбанистики Уральской государственной архитектурно-художественной академии (зав. каф., канд. физ.-мат. наук А.И. Кривоногов); начальник группы отд. 315 НПО «Автоматика», с. н. с., канд. техн. наук Г.П. Лосев

Авторы: В.Г. Лисиенко, Н.Г. Дружинина, О.Г. Трофимова, С.П. Трофимов

М74 Моделирование систем с использованием информационных технологий: учебн. пособие / В. Г. Лисиенко, Н. Г. Дружинина, О. Г. Трофимова, С. П. Трофимов. Екатеринбург: УГТУ-УПИ, 2009. 440 с.

ISBN 978-5-321-01604-6

Учебное пособие предназначено для специалистов, занимающихся вопросами моделирования сложных технических систем и разработки Интернет-приложений, в первую очередь для студентов, обучающихся по направлениям 220200 – Автоматизация и управление и 230100 – Информатика и вычислительная техника. Пособие будет полезно всем, кто связан с исследованием транспортных систем городов, занимается проблемами урбанистики, а также студентам и аспирантам высших учебных заведений.

Библиогр.: 57 назв. Табл. 28. Рис. 233.

Подготовлено кафедрами «Автоматика и управление в технических системах» и «Автоматика и информационные технологии»

УДК 004.94(075.8)
ББК 32.973.26-018я73

ISBN 978-5-321-01604-6

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данное учебное пособие посвящено весьма актуальной и в то же время чрезвычайно трудной проблеме моделирования сложных систем. Известно, что понятие «сложная система» весьма разнообразно и охватывает различные по своей природе явления и технологии, в частности транспортные задачи.

Учебное пособие ставит своей целью, с одной стороны, ознакомить студентов с многообразием алгоритмов и программных средств, используемых при анализе сложных систем, а с другой – на конкретных примерах транспортных задач продемонстрировать реальное применение рассмотренных средств и показать практическую значимость и ценность решаемых транспортных проблем, особенно применительно к среде большого городского мегаполиса.

В соответствии с основной концепцией учебного пособия в первой главе рассмотрены популярная инструментальная среда разработки приложений *Delphi* и язык программирования Интернет-приложений *PHP*. Описаны ключевые моменты объектно-ориентированного программирования. Приведены примеры разработки приложений в среде *Delphi,* веб-сайта с помощью *PHP*. Рассмотрены возможности языка *SQL*, реализованные в СУБД *MySQL*. Основной акцент сделан на разработке приложений с использованием баз данных.

Во второй главе в качестве инструментальной среды моделирования описываются языки имитационного моделирования *GPSS, VISSIM, Simulink MatLab* и авторская разработка *SIANRG*. Изложены теоретические основы имитационного моделирования систем массового обслуживания на языке GPSS. Подробно рассмотрен пример моделирования автоматизированных информационных систем – от построения концептуальной модели системы до ее машинной реализации на языке *GPSS*. Приведены примеры моделирования системы управления, представленной дифференциальными уравнениями с помощью пакетов *VISSIM, Simulink MatLab*, а также системы автоматического регулирования, описанной передаточными функциями. Представлен анализ качества характеристик системы при варьировании параметров объекта. С помощью программного модуля *SIANRG* подробно рассмотрено детерминированное моделирование систем автоматического регулирования, основной акцент сделан на определении параметров качества системы при вариациях параметров регулятора. При стохастическом моделировании систем автоматического регулирования анализируется чувствительность модели при случайных вариациях параметров. Обработка и анализ результатов стохастического моделирования систем автоматического управления представлены с помощью корреляционного, регрессионного и дисперсионного анализов.

Основным объектом анализа и моделирования третьей главы является такая сложная среда, как транспортная система мегаполиса. Дан обзор

специализированных сред разработки приложений для моделирования транспортных и пассажирских сетей города. Приведены примеры информационно-коммуникативных систем российских и зарубежных городов. Описана авторская разработка системы моделирования сети пассажирского транспорта абстрактного города, для которой предлагается ряд критериев оценки качества сети и графика движения пассажирского транспорта. В качестве примера решена важная задача оптимизации передвижения пассажиров по городу. Для увеличения пропускной способности дорожной системы предложена авторская интеллектуальная система управления движением транспорта на перекрестках.

В четвертой главе рассмотрена оригинальная информационно-коммуникативная система МУП «Трамвайно-троллейбусное управление» г. Екатеринбурга. Описаны концептуальные модели и упрощенные схемы базы данных отдельных блоков системы, разработанной с помощью инструментальной среды *Delphi*. Результаты работы информационно-коммуникативной системы представлены в виде отчетов на информационном сайте «Транспорт города Екатеринбурга». Сайт разработан в виде Интернет-приложения на языке *PHP* с использованием базы данных *MySql*.

Особой ценностью пособия является наличие многочисленных заданий для самостоятельной работы студента.

Учебное пособие предназначено для студентов, обучающихся по направлениям 220200 – Автоматизация и управление и 230100 – Информатика и вычислительная техника. Учебное пособие может быть использовано при изучении целого ряда дисциплин. Например, «Моделирование систем», «Информационное обеспечение систем управления», «Инструментальные средства сетевого программирования».

1. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ СИСТЕМ

1.1. Разработка приложений в среде Delphi. Компоненты Delphi. Свойства и методы

1.1.1. Среда Delphi

Delphi представляет собой не только инструмент разработки приложений, но и сложную среду программирования. Создание прикладных программ, или приложений, *Delphi* осуществляется в интегрированной среде разработки *IDE* (*Integrated Development Environment*). Интегрированная среда разработки *Delphi* представляет собой многооконную систему. После загрузки интерфейс *Delphi* выглядит так, как показано на рис. 1.1.

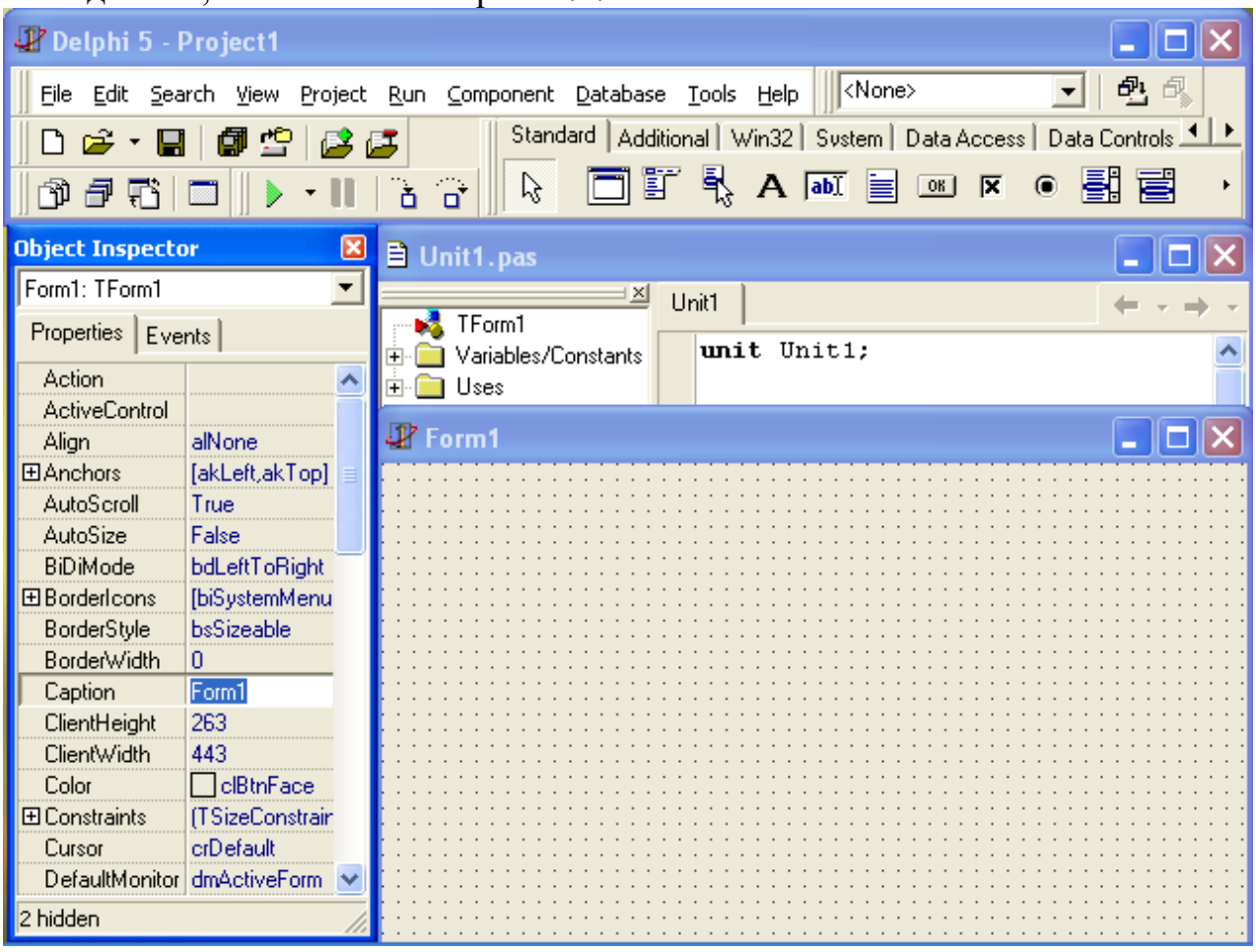


Рис. 1.1. Вид интегрированной среды разработки

Интерфейс *Delphi* первоначально имеет четыре окна:

- главное окно (*Delphi – Project*1);
- окно *Инспектор объектов* (*Object Inspector*);
- окно *Конструктор формы* (*Form*1);
- окно *Редактор кода* (*Unit1.pas*) и др.

В главном окне *Delphi* отображаются:

- Главное меню;
- Панели инструментов;
- Палитра инструментов.

Главное меню содержит обширный набор команд для доступа к функциям *Delphi*.

Панель инструментов находится под главным меню в левой части главного окна и содержит 15 кнопок для вызова часто используемых команд главного меню, например *File/Open* (Файл/Открыть) или *Run/Run* (Выполнение/Выполнить).

Всего имеется 5 панелей:

- *Standard* (Стандартная);
- *View* (Просмотр);
- *Debug* (Отладка);
- *Custom* (Пользователь);
- *Desktop* (Рабочий стол).

Палитра компонентов находится под главным меню в правой части главного окна и содержит множество компонентов, размещаемых в создаваемых формах. Компоненты являются своего рода строительными блоками, из которых конструируются формы приложения. Все компоненты располагаются на отдельной вкладке (странице), а сами компоненты представлены соответствующими значками (пиктограммами). Первоначально Палитра компонентов имеет следующие вкладки:

- *Standard* – Стандартная;
- *Additional* – Дополнительная;
- *Win*32 – 32-разрядный интерфейс *Windows*;
- *System* – Доступ к системным функциям и др.

1.1.1.1. Характеристика проекта

Приложение, создаваемое в среде *Delphi*, состоит из нескольких элементов, объединенных в проект. В состав проекта входят следующие элементы (в скобках указаны расширения имен файлов):

- Код проекта (*DPR*);
- Описания форм (*DFM*);
- Модули форм (*PAS*);
- Модули (*PAS*);

- Параметры проекта (*OPT*);
- Описание ресурсов (*RES*).

Кроме приведенных файлов автоматически могут создаваться и другие файлы, например их резервные копии: *~DP* – для *DPR*-файлов, *~PA* – для *PAS*-файлов.

1.1.1.2. Файл проекта

При запуске *Delphi* автоматически создается новый проект *Project*1, имя которого отображается в заголовке главного окна *Delphi*. Этот проект имеет в своем составе одну форму *Form*1.

Файл проекта является основным и представляет собой собственно программу. Для приложения, включающего в свой состав одну форму, файл проекта имеет следующий вид:

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Имя программы совпадает с именем проекта и указывается при сохранении этого файла на диске.

Сборка всего проекта выполняется при компиляции файла проекта. При этом имя создаваемого приложения (*EXE*-файла) совпадает с название файла проекта.

В разделе *Uses* указывается имя подключаемого модуля *Forms* и перечисляются подключаемые модули всех форм проекта, первоначально это модуль *Unit*1 формы *Form*1.

Директива $*R* подключает к проекту файл ресурсов, который по умолчанию имеет имя, совпадающее с именем файла проекта, поэтому вместо имени файла ресурса указан символ «*». Кроме этого файла разработчик может подключить к проекту и другие ресурсы, самостоятельно добавив директивы $*R* и указав в них соответствующие имена файлов ресурсов.

Программа проекта содержит всего три оператора, выполняющих инициализацию приложения, создание формы *Form*1 и запуск приложения. При

выполнении разработчиком каких-либо операций с проектом код файла проекта формируется *Delphi* автоматически. Например, при добавлении новой формы в файл проекта добавляются две строки кода, относящиеся к этой форме.

С целью просмотра и редактирования кода файла проекта в окне *Редактор кода* достаточно выполнить команду *Project/View Source* (Проект/Просмотр источника).

1.1.1.3. Форма проекта

Для каждой формы в составе проекта автоматически создаются файл описания (*DFM*) и файл модуля (*PAS*). *Файл описания формы* содержит характеристики формы и ее компонентов. Разработчик обычно самостоятельно управляет этим файлом, используя окно *Конструктор формы* и *Инспектор объектов*. Содержимое файла описания формы определяет ее вид. Оно доступно через Конструктор формы. При необходимости можно отобразить этот файл на экране в текстовом виде. Для этого нужно закрыть окно Конструктора той формы, для которой выполняется отображение файла описания, после чего по команде *File/Open* (Файл/Открыть) файл описания формы открывается в окне. Редактор кода и его содержимое доступно для просмотра и редактирования. Ниже приведен пример описания формы, на которой расположена кнопка *Button*1; кроме того, для этой кнопки создан разработчик события *OnClick* (Нажатие):

```
object Form1: TForm1
  Left = 192
  Top = 154
  Width = 451
  Height = 297
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = 64
```

Top = 40

Width = 75

Height = 25

Caption = 'Button1'

TabOrder = 0

OnClick = Button1Click

end

end.

Для того чтобы снова открыть окно *Конструктор формы*, предварительно в Редакторе кода командой меню *File/Close* (Файл/Закрыть) должен быть закрыт файл описания соответствующей формы. Открытие окна *Конструктор формы* выполняется командой *View/Form…* (Просмотр формы). Открывается диалоговое окно *View Form*, в котором можно выбрать нужную форму (рис. 1.2).

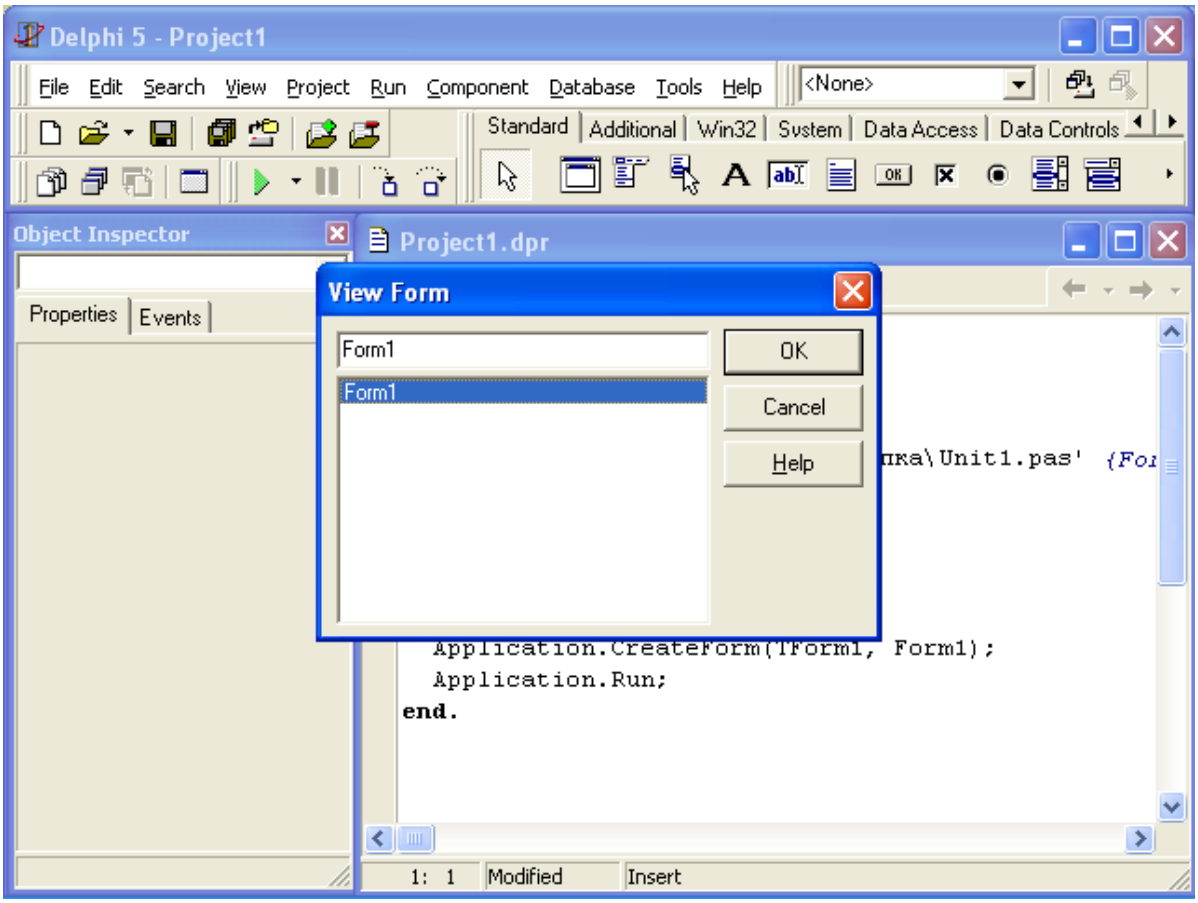


Рис. 1.2. Открытие окна *Конструктор формы*

Файл модуля формы содержит описание класса формы. Для пустой формы, добавляемой к проекту по умолчанию, файл модуля содержит следующий код:

***unit** Unit1;*

interface

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
end.
```

Файл модуля формы создается *Delphi* автоматически при добавлении новой формы. По умолчанию к проекту добавляется форма типа *TForm*, не содержащая компонентов.

В разделе *interface* модуля формы содержится описание класса формы, а в разделе *implementation* – подключение к модулю директивой $*R* описания соответствующей формы. При размещении на форме компонентов, а также при создании разработчиков событий в модуль вносятся соответствующие изменения.

Delphi относится к системам визуального программирования. В процессе разработки в форму помещают компоненты и для них устанавливаются необходимые свойства и обработчики событий.

1.1.1.4. Интуитивный помощник написания кода

Редактор кода *Delphi* оснащен набором средств, обеспечивающих выполнение целого ряда вспомогательных функций. Эти средства имеют общее название *Code Insight – Интуитивный помощник написания кода* (Insight – интуиция). *Code Insight* имеет пять составляющих:

- дополнение кода (*Code Completion*);
- контекстный список параметров (*Code Parameters*);
- быстрая оценка значения (*Tooltip Expression Evaluation*);
- всплывающая подсказка об объявлениях идентификаторов (*Tooltip Symbol Insight*);

- шаблоны кода (*Code Templates*).

Посредством функции дополнения кода в программу могут быть легко введены имена свойств, методов и обработчиков событий объектов. Сначала необходимо ввести имя объекта, поставив в конце точку, например: *Edit*1. После некоторой паузы *Delphi* отобразит на экране список всех свойств и методов данного объекта (рис. 1.3).

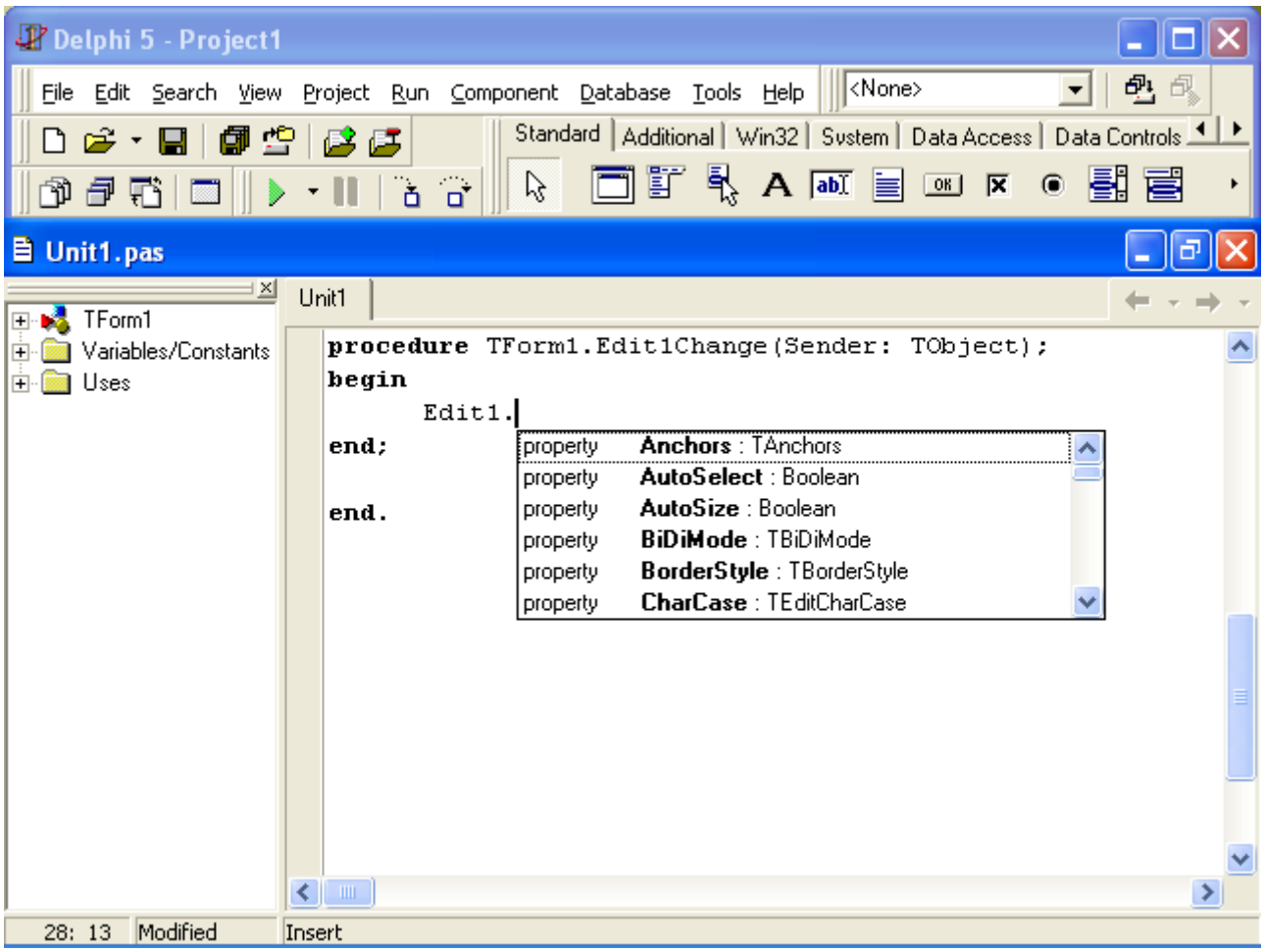


Рис. 1.3. Окно *Code Completion* в окне *Редактор кода*

Шаблоны кода содержат часто используемые программные конструкции, которые можно легко включить в программный код. Шаблоны кода активизируются нажатием клавиш [*Ctrl+J*]. Для целенаправленного поиска шаблона следует сначала ввести ключевое слово (например, *If* или *array*), а затем воспользоваться сочетанием клавиш [*Ctrl+J*]. После этого на экране отобразится список искомых шаблонов кода.

После ввода имени процедуры и открывающейся скобки функция контекстного списка параметров выводит на экране справочное окно (*Hint*) со списком параметров функции или метода.

Функция быстрой оценки кода позволяет проверить значения переменных и свойств. Если в режиме отладки приостановить выполнение приложения и установить курсор на имени переменной или свойства в окне редактора кода,

через некоторое время на экране появится окно с текущим значением этой переменной.

Всплывающая подсказка об объявлениях идентификаторов отображает на экране всплывающую подсказку о типе и месте объявления идентификатора при установке курсора на его имени в окне редактора кода.

1.1.1.5. Инспектор объектов

При разработке приложения часто приходится использовать *Инспектор объектов* (*Object Inspector*). Окно *Object Inspector* содержит две страницы, каждую из которых можно активизировать, выполнив щелчок на вкладке с соответствующим названием. Первая страница имеет название *Properties*. Левая колонка этой страницы содержит список всех свойств редактируемого компонента, доступных во время редактирования. Вторая колонка называется *Events*. В ее левой колонке перечислены все имеющиеся обработчики событий компонента. В правых колонках обеих страниц могут устанавливаться значения соответствующих свойств или обработчиков событий. На рис. 1.1 изображены страницы *Properties* и *Events* инспектора объектов для новой формы.

1.1.2. Примеры создания приложения и задания для самостоятельной работы

Пример 1

Создать новую форму. В форму поместить компоненты: кнопки *Button*1, *Button*2, *Edit*1, *Edit*2, *Edit*3, *ListBox*1.

Внешний вид компонента определяется его свойствами, которые доступны в окне *Инспектор объектов*, когда компонент выделен на форме и вокруг него отображаются маркеры выделения. Выберем свойство *Caption* и изменим заголовок кнопки «Закрыть».

Для того чтобы кнопка могла реагировать на какое-либо событие, необходимо создать или указать процедуру обработки события, которая будет вызываться при возникновении данного события. Поскольку при нажатии на кнопку возникает событие *OnClick*, следует создать обработчик именно этого события. Для этого нужно в *Инспекторе объектов* (вкладка *Events*) сделать двойной щелчок в области значения события *OnClick* и написать код – *Close*:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   close;
end;
procedure TForm1.Button2Click(Sender: TObject);
var i:integer;
```

```
begin
    Edit1.Text:= IntToStr(Form1.ComponentCount);
    ListBox1.Clear;
    for i:=0 to   Form1.ComponentCount-1 do
    begin
        ListBox1.Items.add(IntToStr(i+1)+'.'+Form1.Components[i].Name);
    end;
end;
```

Для кнопки *Button*2 напишем программный код, выводящий список всех компонентов формы в список *ListBox*1. Результат выполнения события показан на рис. 1.4.

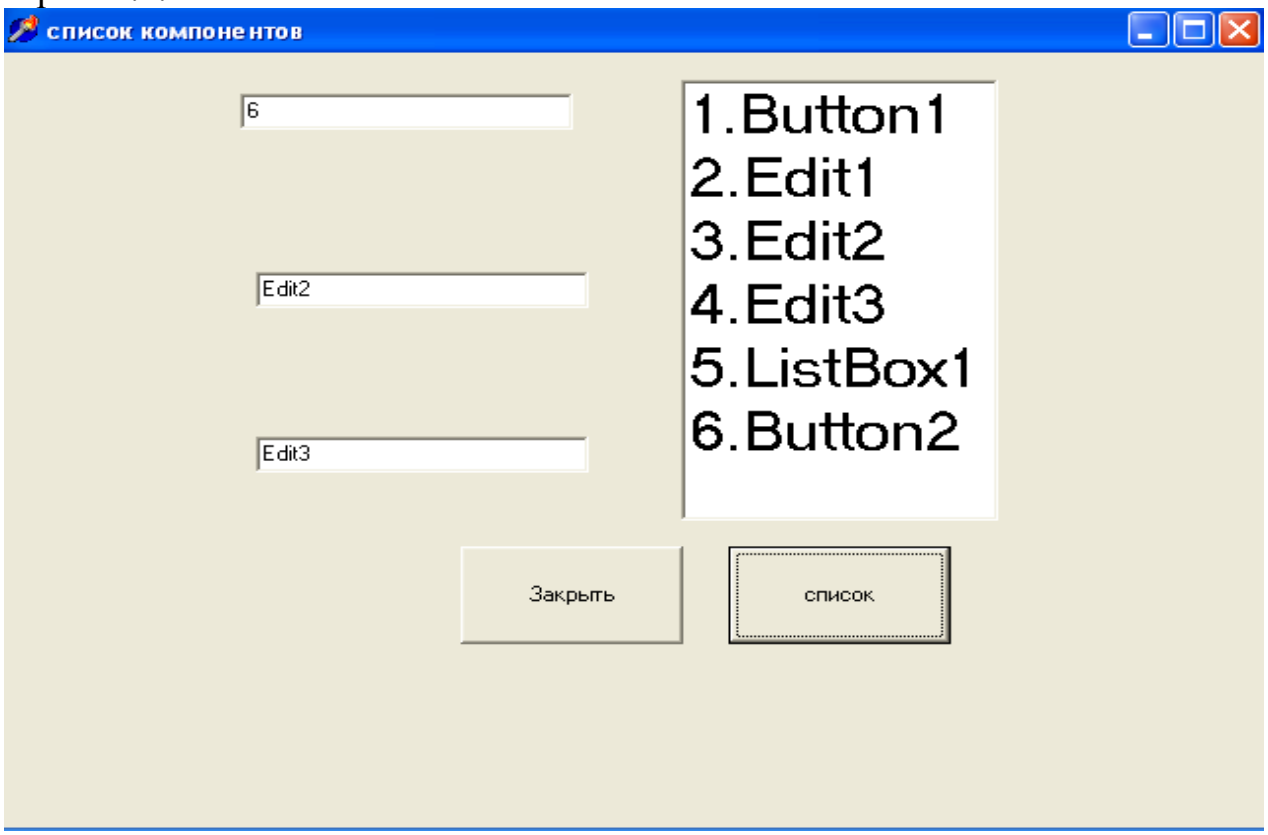


Рис. 1.4. Результаты процедуры *Button*2*Click* – заполненный список *TListBox*

Задание. Используя компонент *OpenColor*1, измените фон компонента *ListBox*1, цвет выводимого списка по выбору из предложенного списка компонента *OpenColor*1.

Пример 2

Компонент *TSplitter* – это компонент страницы *Additional* палитры компонентов. *TSplitter* позволяет изменять размеры элементов управления непосредственно во время выполнения приложения путем перетаскивания разделительных полос.

Для того чтобы использовать в приложении компонент *TSplitter*, необходимо:

- поместить элемент управления (например, *TMemo*) в форму и выровнять его по одной из сторон формы, присвоив свойству значение *AlLeft* или *AlRIght*;
- поместить в форму компонент *TSplitter* и присвоить его свойству *Align* то же значение, что и свойству *Align* вставленного ранее элемента управления;
- повторить два предыдущих действия всех элементов управления формы;
- присвоить свойству *Align* последнего элемента управления значение *AlClient* (чтобы этот элемент заполнял все оставшееся в форме место).

Для того чтобы элементы управления, помещенные в форму, можно было уменьшать до определенного размера, следует присвоить соответствующее значение свойству *MinSize*. После запуска проекта форма выглядит так, как показано на рис. 1.5.

Задание. Используя свойства *alTop* и *alButtom*, сделайте разделительные полосы вертикально.

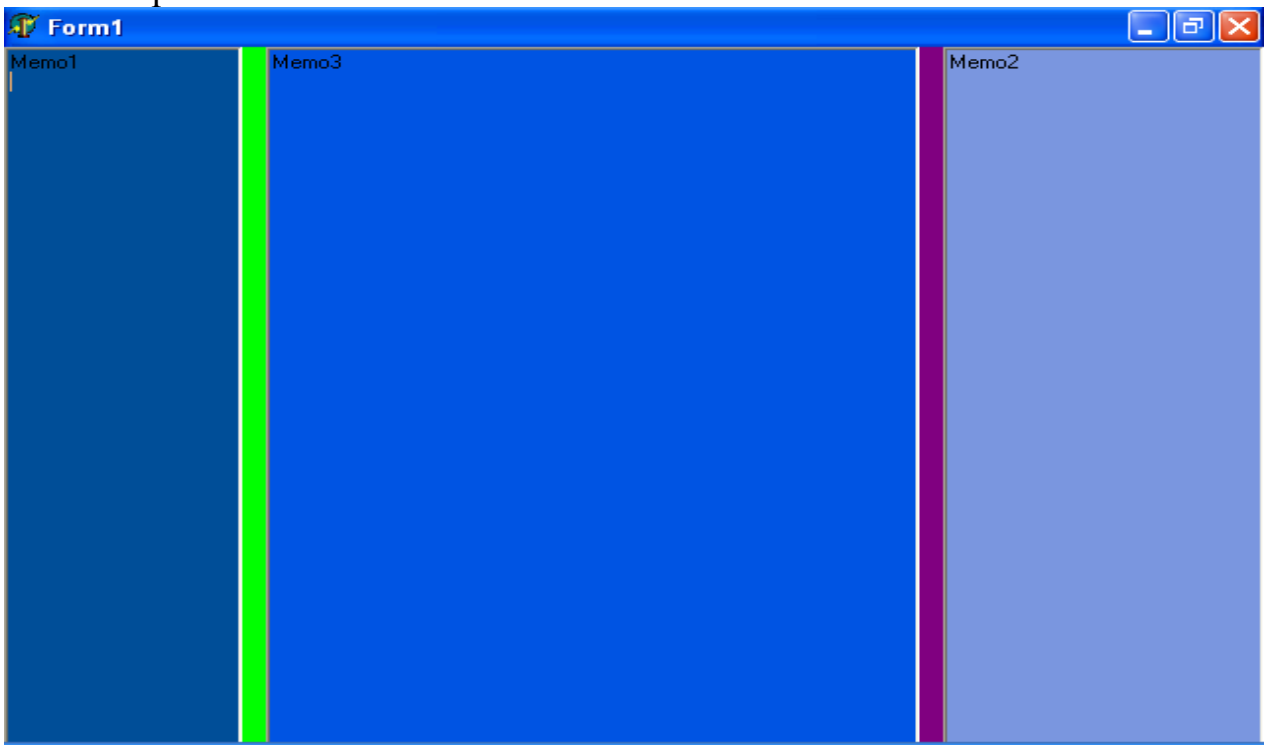


Рис. 1.5. Форма с компонентами *TSplitter* после изменения размера во время выполнения приложения

Пример 3

Расположите в форме компоненты группу опций *RadioGroup*1, линейку прокрутки *ScrollBar*1, кнопку *BitBtn*1. Для *BitBtn*1 у свойства *Kind* выберите *bkClose*. Это свойство определяет, какое изображение и с каким текстом будет отображаться на кнопке *BitBtn*. Когда пользователь выполнит щелчок на опции, расположение линейки прокрутки меняется (см. рис. 1.6):

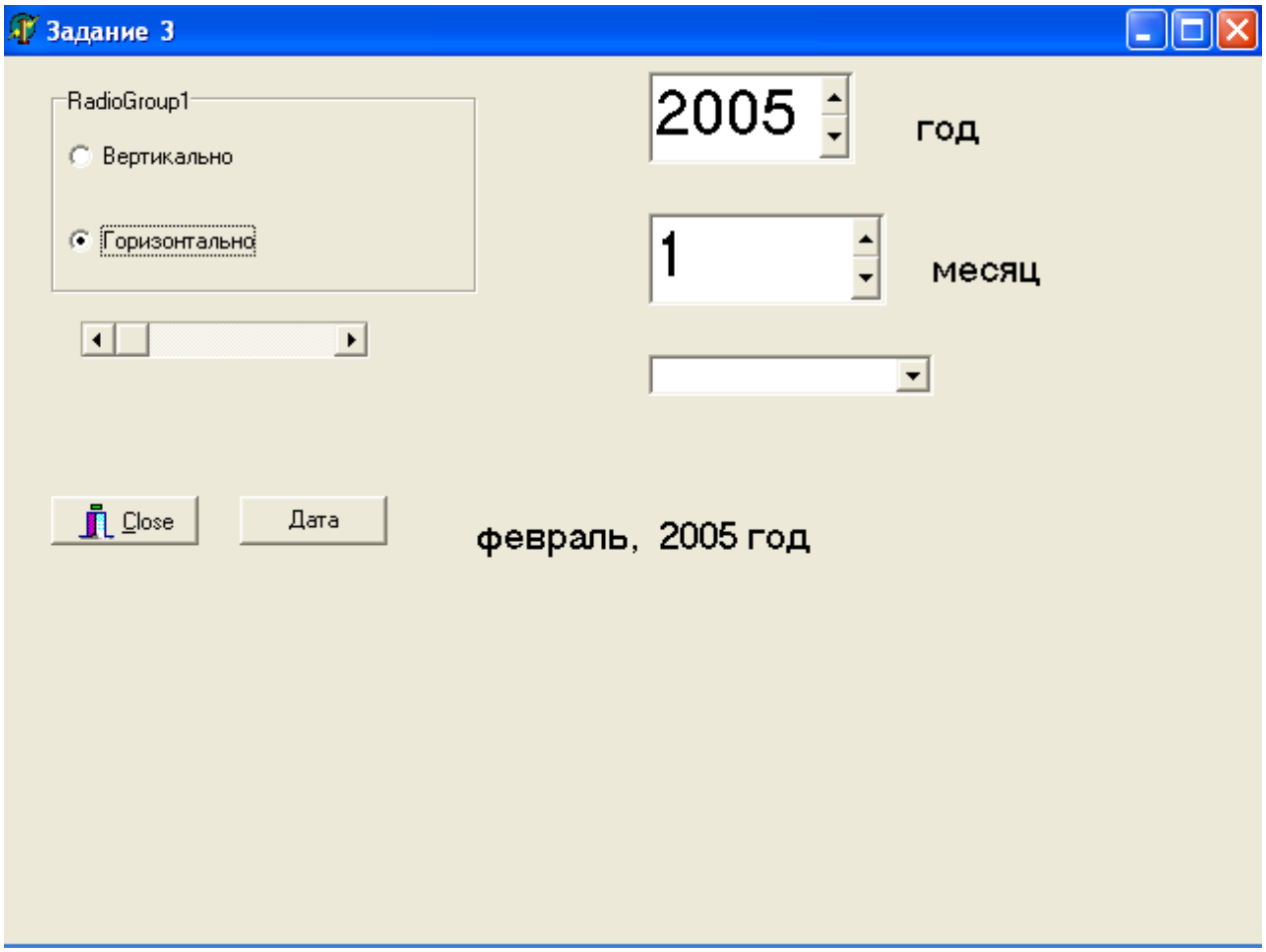


Рис. 1.6. Форма к примеру 3 и заданию 3. Результат нажатия кнопки «Дата»

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    RadioGroup1.Items.add('Вертикально');
    RadioGroup1.Items.add('Горизонтально');
    RadioGroup1.ItemIndex:=2;
end;
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
    if RadioGroup1.Items[RadioGroup1.ItemIndex]= 'Вертикально'
    then
    ScrollBar1.Kind:=sbVertical;
    if RadioGroup1.Items[RadioGroup1.ItemIndex]= 'Горизонтально'
    then
    ScrollBar1.Kind:=sbHorizontal;
end.
```

Задание. В форму поместите компоненты *SpinEdit*1, *SpinEdit*2, *Label*1, *Label*2, *ComboBox*1. Свойству *Value* компонента *SpinEdit*1 присвойте начальное значение 2005. Свойству *Value* компонента *SpinEdit*2 присвойте начальное значение 1. У *Label*1 подпишите – год, у *Label*2 – месяц. В *ComboBox*1 перечислите названия месяцев года. Добавьте кнопку. Присвойте ей название – Дата. По щелчку на эту кнопку в компоненте *Label*3 должно появиться название месяца, соответствующее значению переменной в компоненте *SpinEdit*2, а год – значению в компоненте *SpinEdit*1. Изменяя значения в компонентах *SpinEdit*1, *SpinEdit*2, меняем текстовое название месяца в компоненте *Label*3. В компоненте *SpinEdit*2 установите ограничение на выбор максимального числа месяца 12. Результат показан на рис. 1.6.

Пример 4

Метод *function ItemRect* (*Item: integer*): *TRect* возвращает координаты прямоугольника, ограничивающего указанный в параметре *Item* элемент. В форме содержится список и четыре метки (*TLabel*). При создании формы в список включаются три строки. Когда пользователь выбирает одну из строк, в компонентах *TLabel* отображаются координаты прямоугольника, который ограничивает выбранную строку. По кнопке *Add* добавим какие-нибудь значения в список. Отсортируем его (результат выполнения смотри на рис. 1.7).

```
procedure TForm1.ListBox1Click(Sender: TObject);
var   ListBoxItem:TRect;
begin
    ListBoxItem:=ListBox1.ItemRect(ListBox1.ItemIndex);
    Label1.Caption:='Слева '+ IntToStr(ListBoxItem.Left);
    Label2.Caption:='Сверху '+ IntToStr(ListBoxItem.Top);
    Label3.Caption:='Справа '+ IntToStr(ListBoxItem.Right);
    Label4.Caption:='Снизу '+ IntToStr(ListBoxItem.Bottom);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
    with ListBox1 do
    begin
        items.add('Привет') ;
        items.add('мы вместе') ;
        items.add('до свидания') ;
    end;
procedure TForm1.addClick(Sender: TObject);
```

```
begin
    ListBox1.Items.Add(Edit1.Text);
end;
procedure TForm1.SortClick(Sender: TObject);
begin
    ListBox1.Sorted:=True;
end;
```

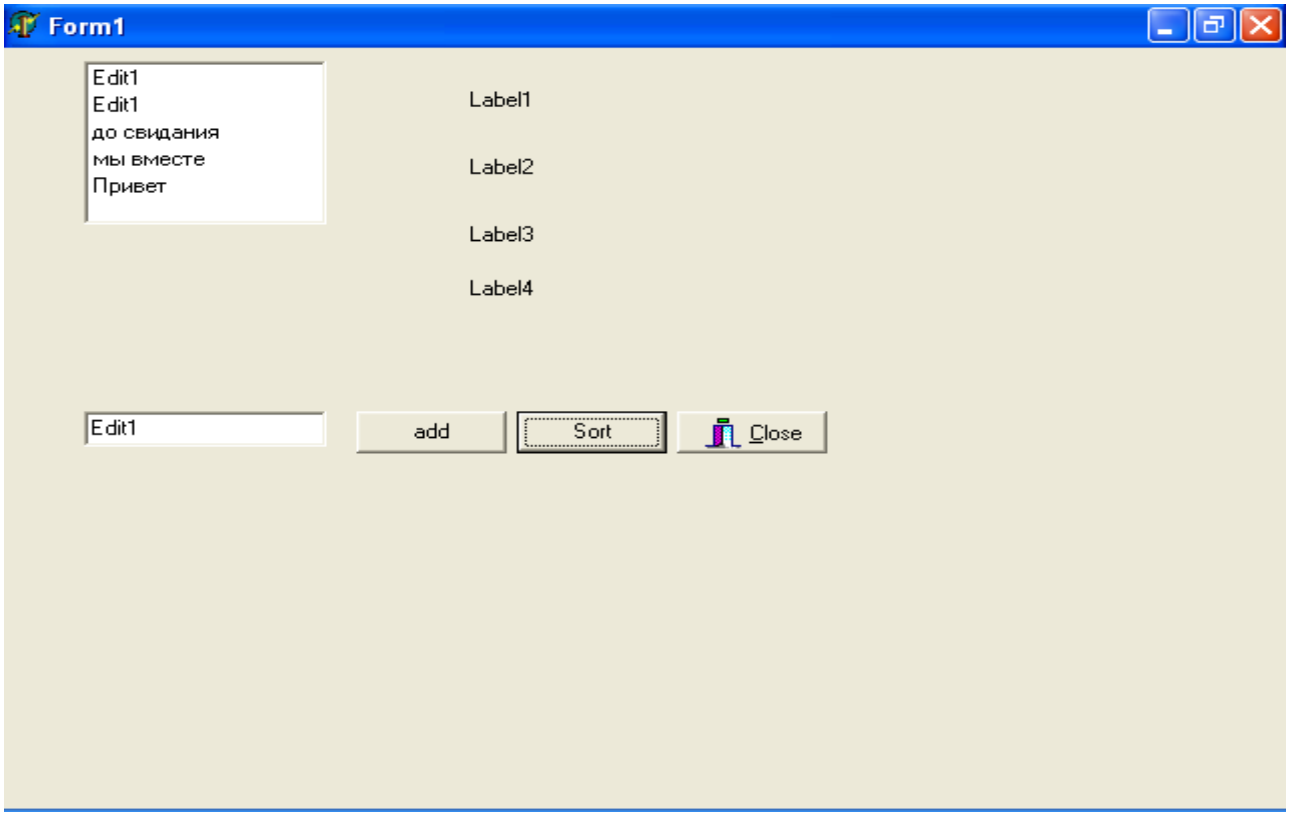


Рис. 1.7. Форма примера 4 – результат обработки события на кнопке *Sort*

Задание. Откройте новую форму. Поместите компоненты: *Edit*1, *Edit*2, *Edit*3, *Edit*4, *Button*1, *Button*2, *Button*3, *ListBox*1. Для кнопки *Button*1 напишите процедуру закрытия формы. Для кнопки Button1 напишите процедуру решения уравнения, например уравнения $Ax^2 + By^2 = C$. Уравнение задается пользователем в компоненте *Edit*1. Параметры *A*, *B*, *C* задаются в других компонентах *Edit* соответственно. Результат заносится в *ListBox*1. Для *Button*3 напишите процедуру сортировки получившихся значений.

Пример 5

Свойство *ReadOnly* определяет, может ли пользователь изменить текст элементов управления. Следующая процедура изменит состояние *ReadOnly* поля ввода с *True* на *False* и наоборот с *False* на *True*, когда пользователь выполнит двойной щелчок на форме (результат выполнения смотри на рис. 1.8).

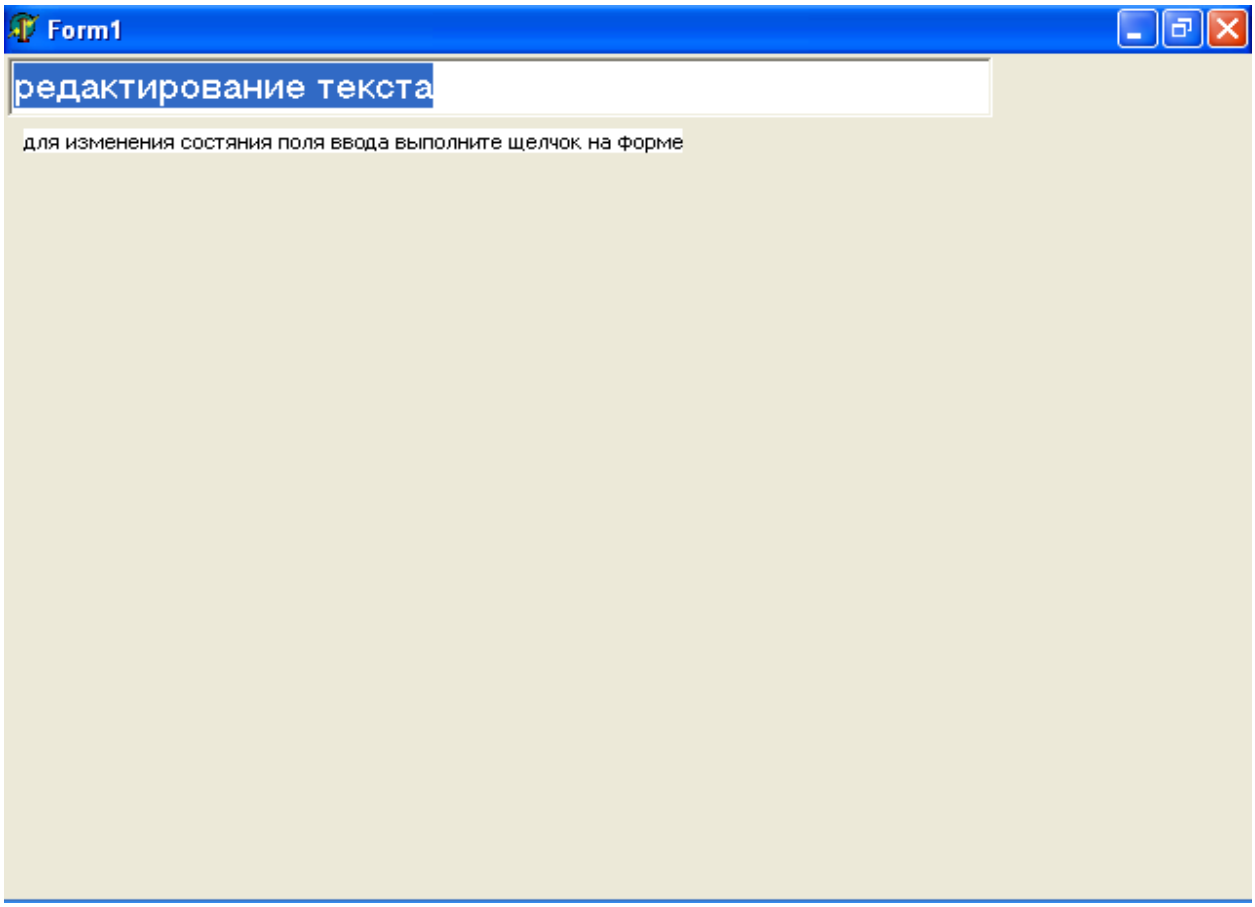


Рис. 1.8. Результат выполнения примера 5

```
procedure TForm1.FormActivate(Sender: TObject);
begin
    Edit1.Left:=2;
    Edit1.Top:=2;
    Edit1.ReadOnly:=true;
    Edit1.Text:='редактирование текста';
end;
procedure TForm1.FormDblClick(Sender: TObject);
begin
    Edit1.ReadOnly:= not Edit1.ReadOnly;
end;
procedure TForm1.FormPaint(Sender: TObject);
begin
    Canvas.TextOut(10,40,'для изменения состояния поля ввода '+
    'выполните щелчок на форме')
end;
```

Задание. Добавьте в форму компоненты *TMemo,TButton*. Напишите программный код записи всего текста или части текста из *Edit*1 в *Memo*1. Используйте методы *SelectAll, SelStart, SelLenth*. Добавьте в форму *Edit*2, *Edit*3. Укажите в новой строке поля *Memo*1 количество знаков и с какого знака скопирована часть текста по выбору пользователя.

Пример 6

Компонент *TAnimate* предназначен для воспроизведения видеофильмов. В свойстве *CommonAVI* выберите любой *AVI*-клип. Для кнопки напишите следующую процедуру (результат показан на рис. 1.9):

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Animate1.Play(Animate1.startFrame,Animate1.StopFrame,0);
end;
```

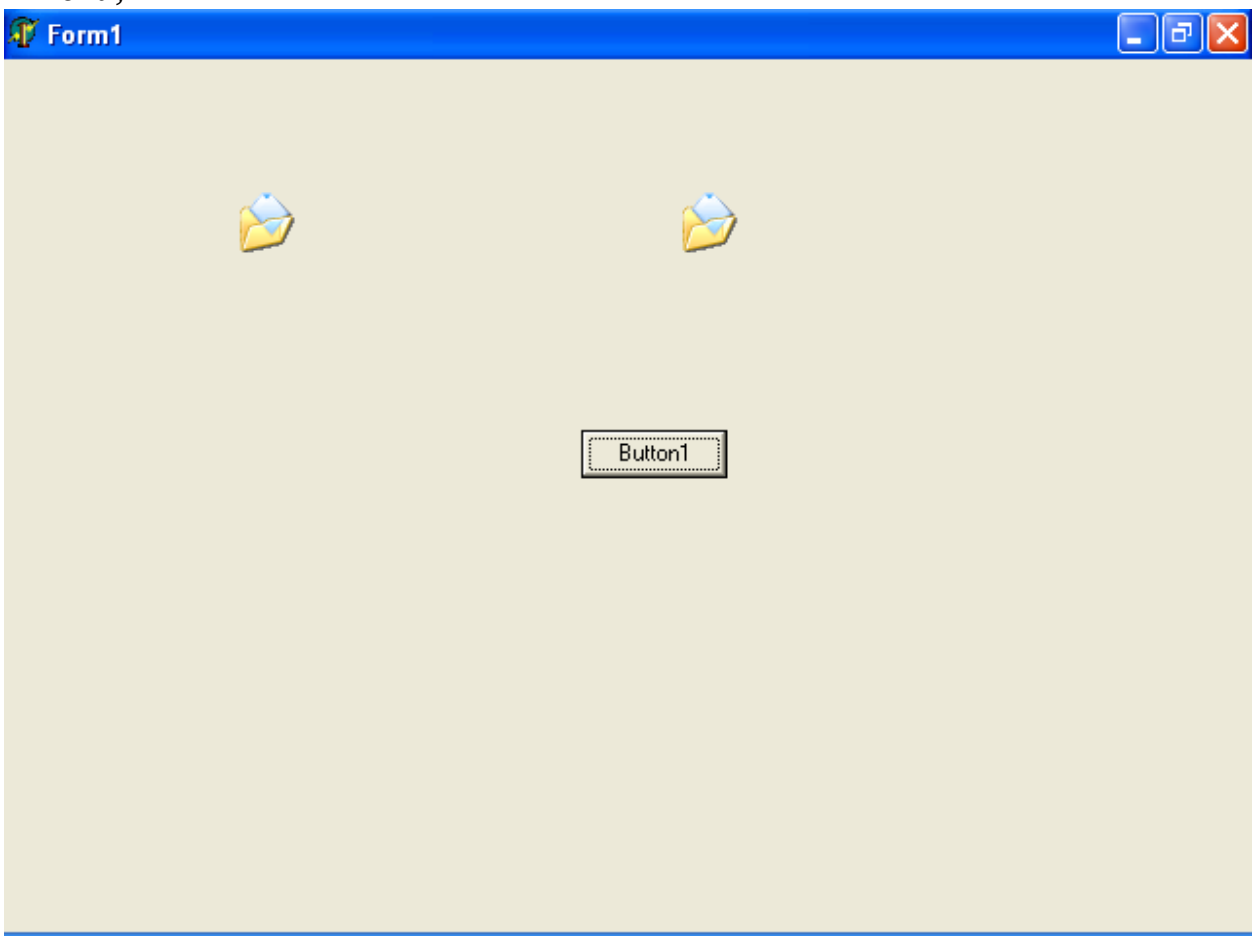


Рис. 1.9. Работа компонента *Animate*1

Задание. Добавьте необходимые компоненты *(Edit*1, *Edit*2) и отредактируйте программный код так, чтобы пользователь мог сам управлять параметрами компонента *TAnimate*. Не забудьте всем кнопкам дать названия, понятные для пользователя.

Пример 7

Форма содержит компонент *TDateTimePicker* и кнопку. Свойство *Kind* имеет значение *dtkTime*. Когда пользователь выполняет щелчок на кнопке, появляется окно сообщения, в котором отображается текущее время (результат выполнения процедуры *Button1Click* показан на рис. 1.10):

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    MessageDlg('Текущее время:'+ TimeToStr(DateTimePicker1.Time),
    mtInformation,[mbOk],0);
end;
```

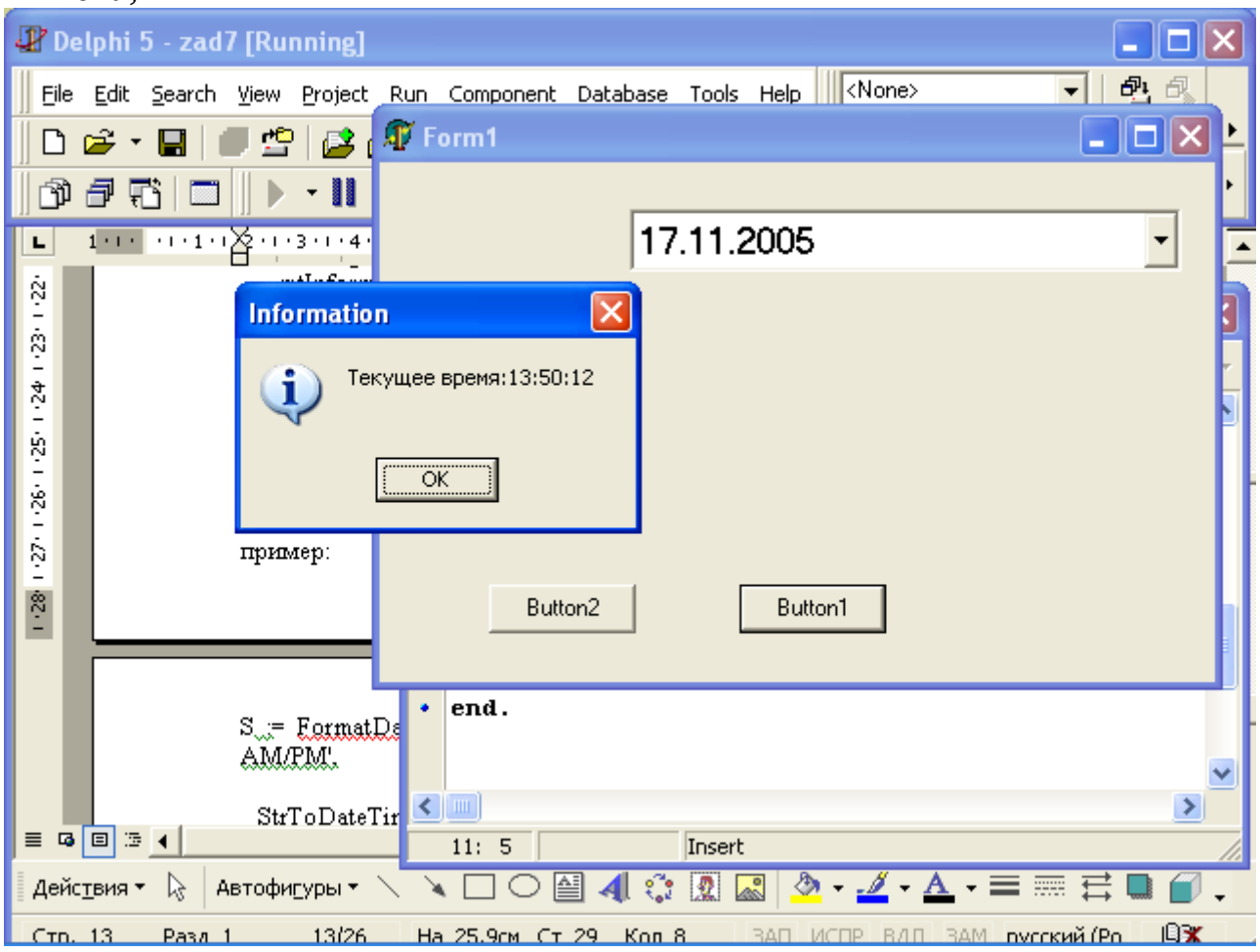


Рис. 1.10. Текущее время во время выполнения приложения

Задание. Поменяйте свойство *Kind* на *dtkDate*. Установите текущую дату при открытии формы, используя функцию *Now*(). В компоненте списков заполните список месяцев года. По кнопке в компоненте выделите месяц, выбранный на календаре. Используйте функцию *FormatDateTime* для определения номера месяца:

function *FormatDateTime(const Format: string; DateTime: TDateTime): string;*

Например: дата Среда, 15 февраля 2005 года, время 10:30 утра .

S := FormatDateTime('"The meeting is on " dddd, mmmm d, yyyy, " at " hh:mm AM/PM', StrToDateTime('2/15/05 10:30am')).

Пример 8

В данном примере определена процедура *checkState*, которая с помощью *WinApi*-фукции *GetKeyState* проверяет состояние клавиши [*CapsLock*]. Если эта клавиша нажата, в первом разделе строки состояния появляется надпись ЗАГЛАВНЫЕ. Процедура *checkState* связана с обработчиком событий *FormKeyDown*, который вызывается при нажатии клавиши (рис. 1.11):

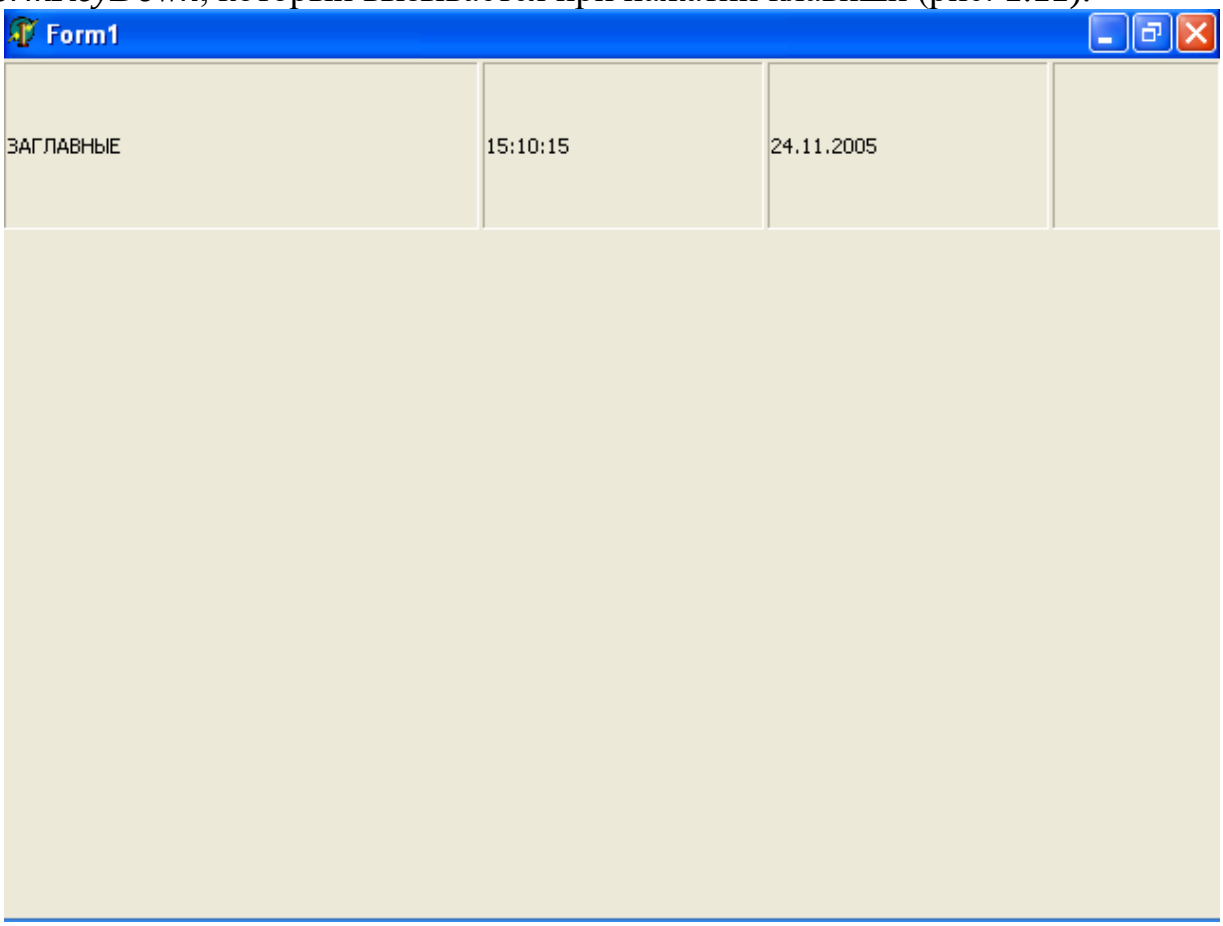


Рис. 1.11. Вид формы после включения клавиши CapsLock

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    checkState;
end;
procedure TForm1.checkState;
begin
  if  odd(GetKeyState(VK_CAPITAL)) then
```

```
    StatusBar1.Panels[0].Text:='ЗАГЛАВНЫЕ'
  else
    StatusBar1.Panels[0].Text:='';
end;
```

Задание. Покажите в разных панелях строки состояния текущую дату, время, режим замены/вставки, включение/выключение клавиши '*Num Lock*'.

Пример 9

Элемент управления *TShape* предоставляет пользователю набор геометрических фигур (примитивно), которые можно поместить в любом месте формы. Следующая процедура изменит форму, цвет и образец заполнения компонента (рис. 1.12):

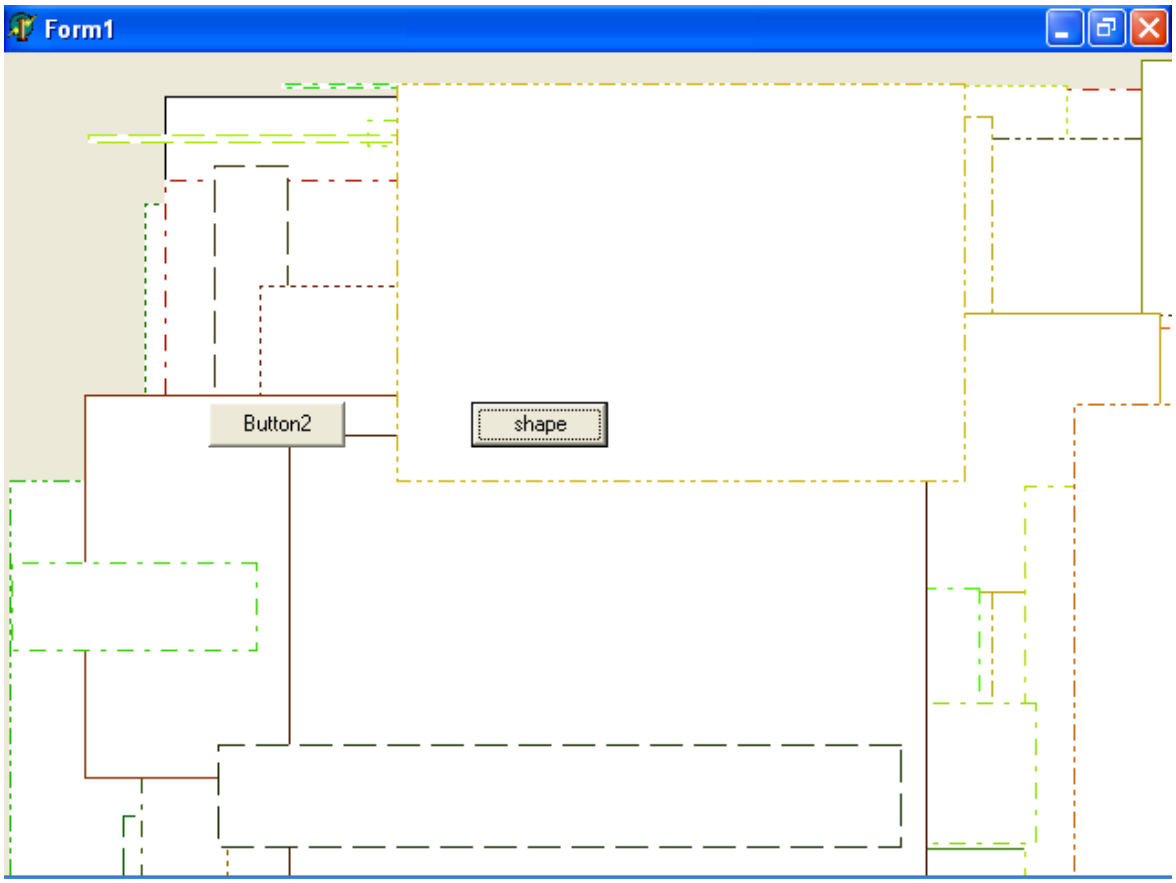


Рис. 1.12. Форма во время выполнения

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Shape1.Shape:=stEllipse;
    Shape1.Brush.Color:=clMaroon;
    Shape1.Brush.Style:=bsBDiagonal;
end;
```

Задание. Добавьте в форму еще один компонент *TShape*, *TTimer*. С помощью свойства *Pen* измените толщину прямоугольника. Добавьте следующий программный код. Запустите программу:

```
var
  X, Y: Integer;
procedure TForm1.FormActivate(Sender: TObject);
begin
  WindowState := wsMaximized;
  Timer1.Interval := 50;
  Randomize;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  X := Random(Screen.Width - 10);
  Y := Random(Screen.Height - 10);
  Canvas.Pen.Color := Random(65535);
  case Random(5) of
    0: Canvas.Pen.Style := psSolid;
    1: Canvas.Pen.Style := psDash;
    2: Canvas.Pen.Style := psDot;
    3: Canvas.Pen.Style := psDashDot;
    4: Canvas.Pen.Style := psDashDotDot;
  end;
  Canvas.Rectangle(X, Y, X + Random(400), Y + Random(400));
end;
```

Пример 10

Рассмотрим работу с компонентом меню *TMenu*. Форма приложения содержит меню *MainMenu*1, компонент *Timer*1. Дизайнер меню активизируется двойным щелчком на соответствующем компоненте меню в форме, показанной на рис. 1.13. С помощью дизайнера меню создайте меню формы. Основные пункты: цвет (подпункты – синий, красный, восстановление цвета формы), управление пунктами меню (время, установка отметки), выход (результат выполнения пунктов меню представлен на рис. 1.14):

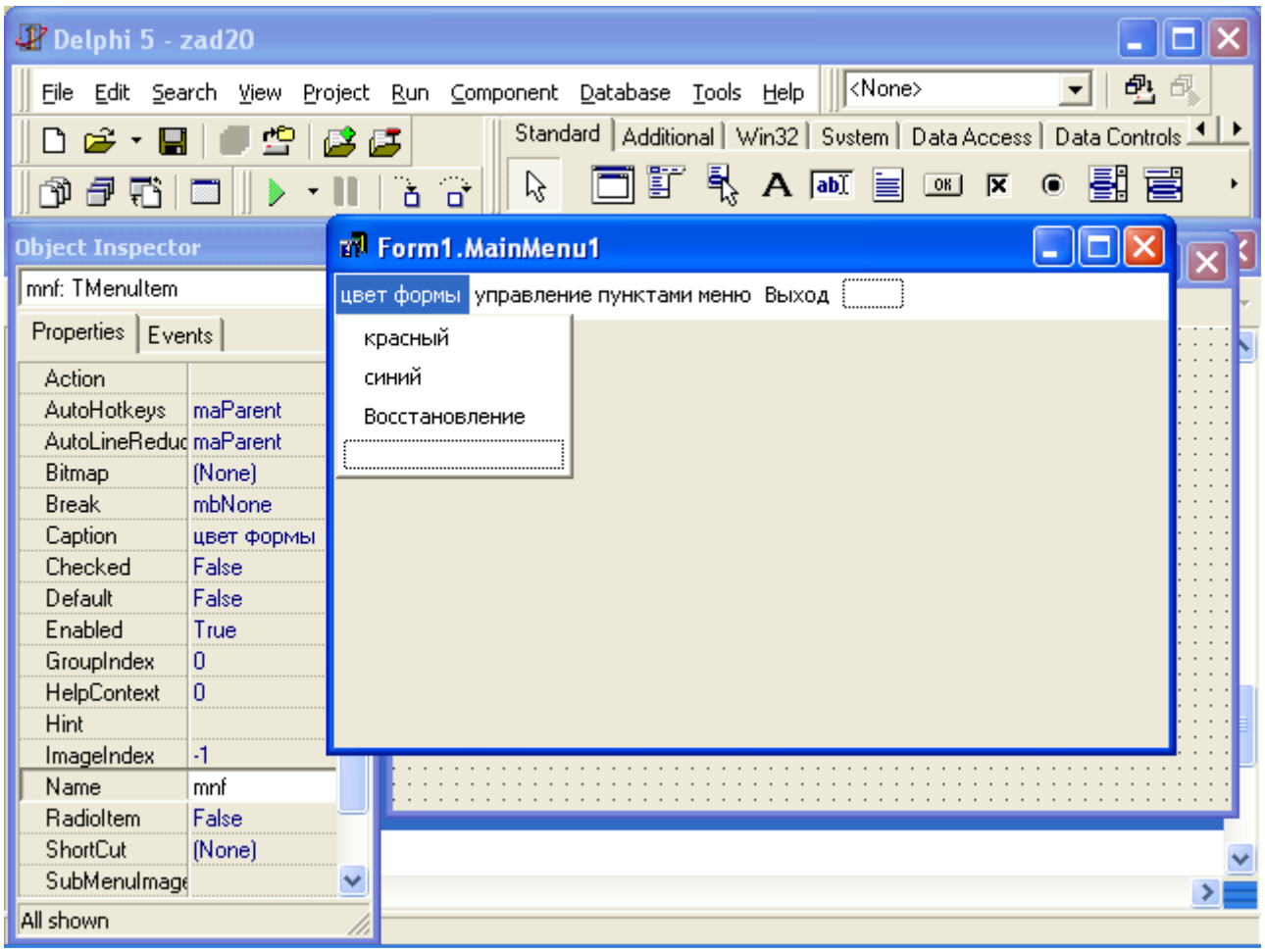


Рис. 1.13. Дизайнер меню

```
unit Unit20;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls;
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    mnf: TMenuItem;
    mnfred: TMenuItem;
    mnfbly: TMenuItem;
    mnfreset: TMenuItem;
    vt1: TMenuItem;
    mncl: TMenuItem;
    Nnmtime: TMenuItem;
```

```
    mnotmrtki: TMenuItem;
    Timer1: TTimer;
    procedure mnfredClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure mnfblyClick(Sender: TObject);
    procedure mnfresetClick(Sender: TObject);
    procedure mnclClick(Sender: TObject);
    procedure NnmtimeClick(Sender: TObject);
    procedure mnotmrtkiClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
     FormColorMemory:LongInt;
implementation
{$R *.DFM}
procedure TForm1.mnfredClick(Sender: TObject);
begin
      Form1.Color:=clRed;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
FormColorMemory:=form1.Color;
end;
procedure TForm1.mnfblyClick(Sender: TObject);
begin
      Form1.Color:=clBlue;
end;
procedure TForm1.mnfresetClick(Sender: TObject);
begin
```

```
    form1.Color:=FormColorMemory;
end;
procedure TForm1.mnclClick(Sender: TObject);
begin
    close;
end;
procedure TForm1.NnmtimeClick(Sender: TObject);
begin
    Nnmtime.Caption:='Время'+TimeToStr(Time);
end;
procedure TForm1.mnotmrtkiClick(Sender: TObject);
begin
    if   mnotmrtki.Checked then
         mnotmrtki.Checked :=false
    else
    mnotmrtki.Checked :=true;
end;
end.
```

Задание. Задайте название формы. Добавьте в форму компонент *PopupMenu*1, *ColorDialog*1. Подключите *PopupMenu*1 к форме в окне *Инспектор объектов* формы. Добавьте пункт «цвет формы». Для этого пункта напишите обработчик события: изменение цвета формы по выбору из *ColorDialog*1. Добавьте пункт второй – восстановление первоначальных параметров формы (цвет, отсутствие метки в пункте "установка метки", время в пункте «время» убрать). Добавьте компонент *Label*1, *Edit*1. Третий пункт *PopupMenu*1 «изменение надписи метки». Обработчик данного события должен содержать код переноса текста из редактора текста *Edit*1 в метку *Label*1.

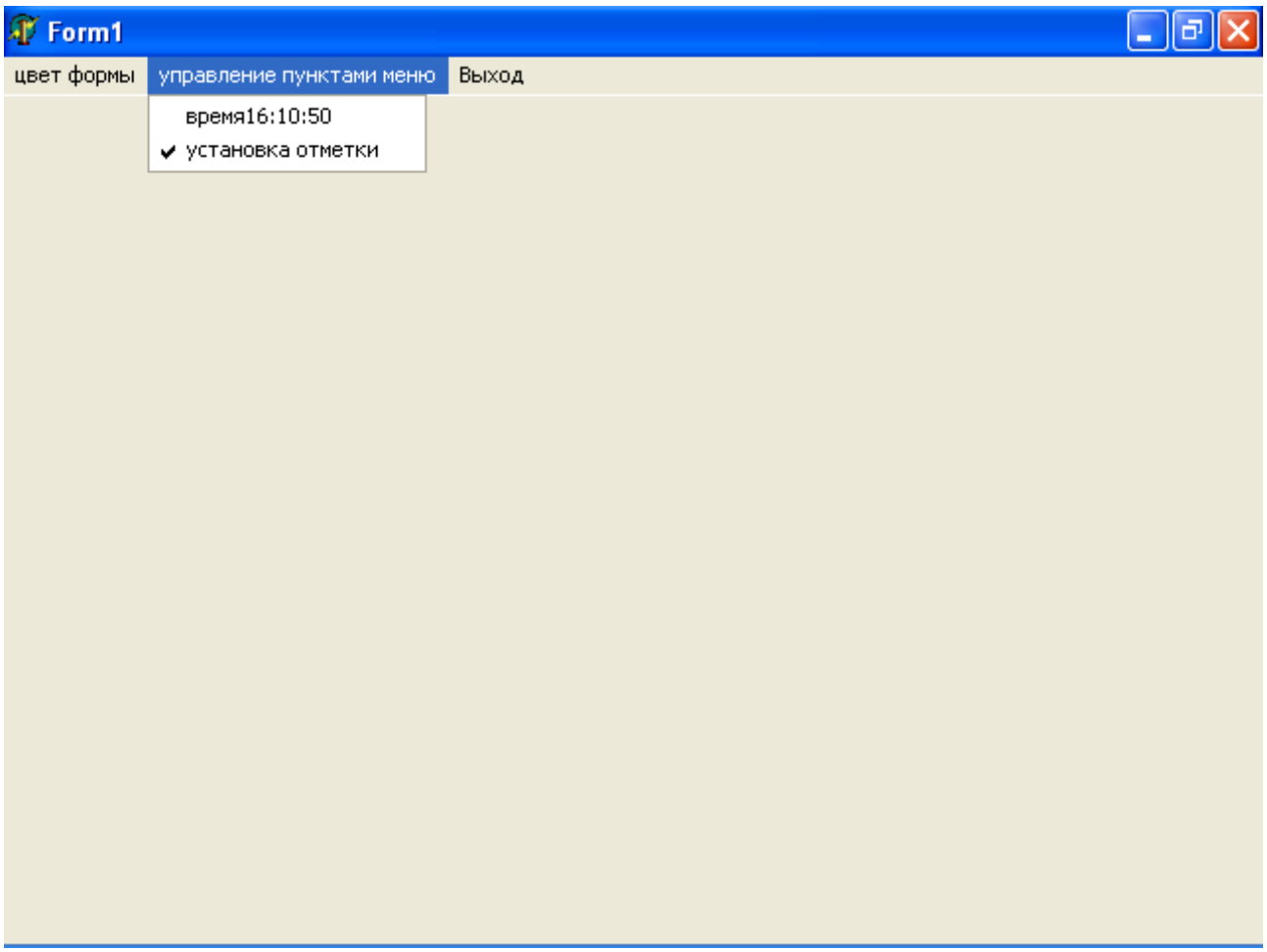


Рис. 1.14. Результат выполнения пунктов меню

1.2. Разработка систем управления базами данных в среде Delphi

1.2.1. События и обработчики событий

Приложения, созданные с помощью *Delphi*, – это приложения для *Windows*. Одним из свойств таких приложений является управление по событиям. Это означает, что программа выполняется на основе генерируемых сообщений о событиях, которые обрабатываются программным кодом приложения. Такой код необходимо написать для каждого события, на которое должна реагировать программа. Процедура, предназначенная для реагирования на какое-либо событие, называется в *Delphi* процедурой обработки события (или *Event Handler*).

1.2.1.1. События

Выделяются две категории событий:

- события, обусловленные действиями пользователя;
- простые или программно-управляемые.

Процедуры обработки пользовательских событий обеспечивают интерактивное взаимодействие приложений и пользователя. В *Delphi* для этой цели применяются предварительно определенные обработчики событий, которые могут использоваться практически всеми компонентами. Сюда относятся обработчики событий *OnClick, OnDbClick,* обработчики событий мыши и клавиатуры.

К обычным событиям относятся события активации, завершения, события изменения состояния компонентов и т. д., которые являются косвенным результатом действия пользователя.

Delphi генерирует процедуры обработки для каждого события и дает им имена в соответствии с именами компонентов, для которых эти процедуры предназначены. Например, в случае обработки события щелчка мышью (*Click*) на элементе управления *Button*1 генерируется пустая процедура обработки события, которая показана на рис. 1.15.

Теперь необходимо вписать нужный программный код между зарезервированными словами *Object Pascal begin* и *end.*

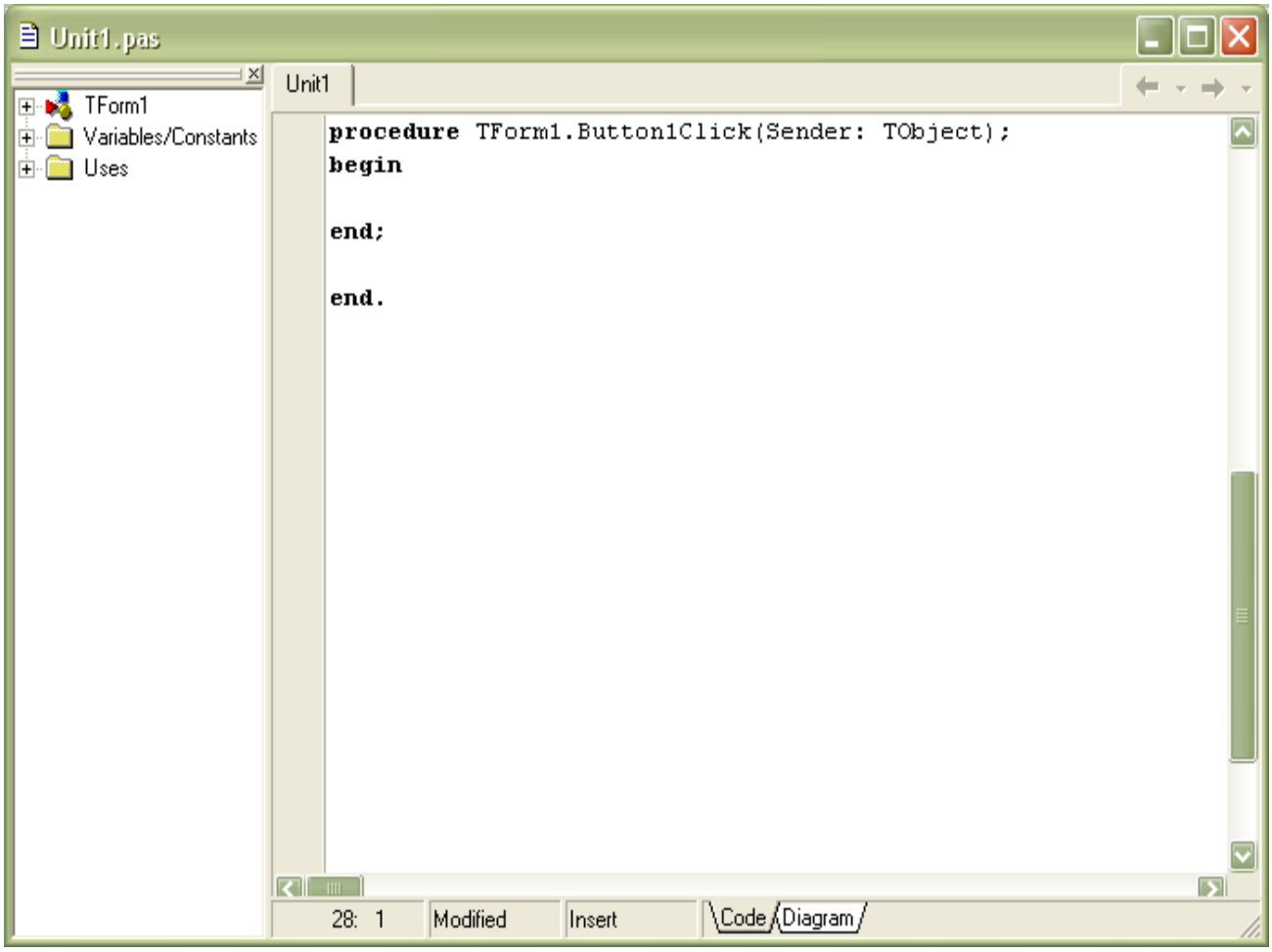


Рис. 1.15. Заголовок процедуры *OnClick* для кнопки *Button*1

1.2.1.2. Действия пользователя с мышью

Когда пользователь передвигает мышь, то по экрану перемещается небольшое изображение. Это называется – указатель мыши. Различают следующие виды действий с мышью, на которые должно реагировать приложение:

- нажатие кнопки мыши (*MouseDown*);
- отпускание кнопки мыши (*MouseUp*);
- перемещение мыши (*MouseMove*);
- щелчок (*Click*);
- двойной щелчок (*DbClick*).

С каждым действием мыши связан определенный обработчик события. Эта информация представлена в виде схемы в табл. 1.1.

Таблица 1.1

Результаты действий пользователя с мышью

Действия пользователя	Состояние кнопки мыши	Обработчик событий мыши
Нажатие кнопки мыши	Кнопка нажата	*OnMouseDown*
Отпускание кнопки мыши	Кнопка не нажата	*OnMouseUp*
Перемещение мыши	Кнопка нажата или не отжата	*OnMouseMove*

1.2.1.3. События клавиатуры

События клавиатуры генерируются как только клавиша будет нажата или отпущена. В обоих случаях приложение получает сообщение от *Windows* о нажатии клавиши. *Windows* посылает в приложение сообщение, состоящее из двух частей: сообщение о нажатии клавиши и сообщение о символе. Это отражено в наличии двух обработчиков событий:

- *OnKeyDown* (вызывается при нажатии клавиши) и *OnKeyUp* (вызывается при отпускании клавиши). Эти события происходят при каждом нажатии клавиши
- *OnKeyPress* – происходит лишь при нажатии клавиши, с которой связан читаемый символ.

1.2.1.4. Перехват событий клавиатуры

Для формы в *Delphi* определено свойство, которое может быть установлено таким образом, что события клавиатуры будут сначала проверяться и обрабатываться самой формой, после чего они могут быть

обработаны активным элементом управления. Это свойство *KeyPreview*. Если данное свойство имеет значение False, события клавиатуры поступают непосредственно к тому элементу управления, который активен для ввода. Если же этому свойству присвоено значение True, события клавиатуры сначала будет получат форма.

Объявление свойства *KeyPreview* выглядит следующим образом:

property KeyPreview: Boolean;

1.2.2. Примеры создания приложения и задания для самостоятельной работы

Пример 1

Обработчики событий *OnMouseDown* и *OnMouseUp* имеют тип:

type *TMouseEvent = procedure (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer) of object;*

В процедуре обработки события следует определить, что должно произойти в качестве реакции на нажатие кнопки мыши. Используйте параметр *Button*, если для разных кнопок мыши должны быть определены разные действия. Компонент *TShape*, имеющий форму круга, создается и отображается при каждом нажатии кнопки мыши. Однако, если нажимается правая кнопка, появляющийся в данной форме круг имеет черный цвет и красный контур (рис. 1.16):

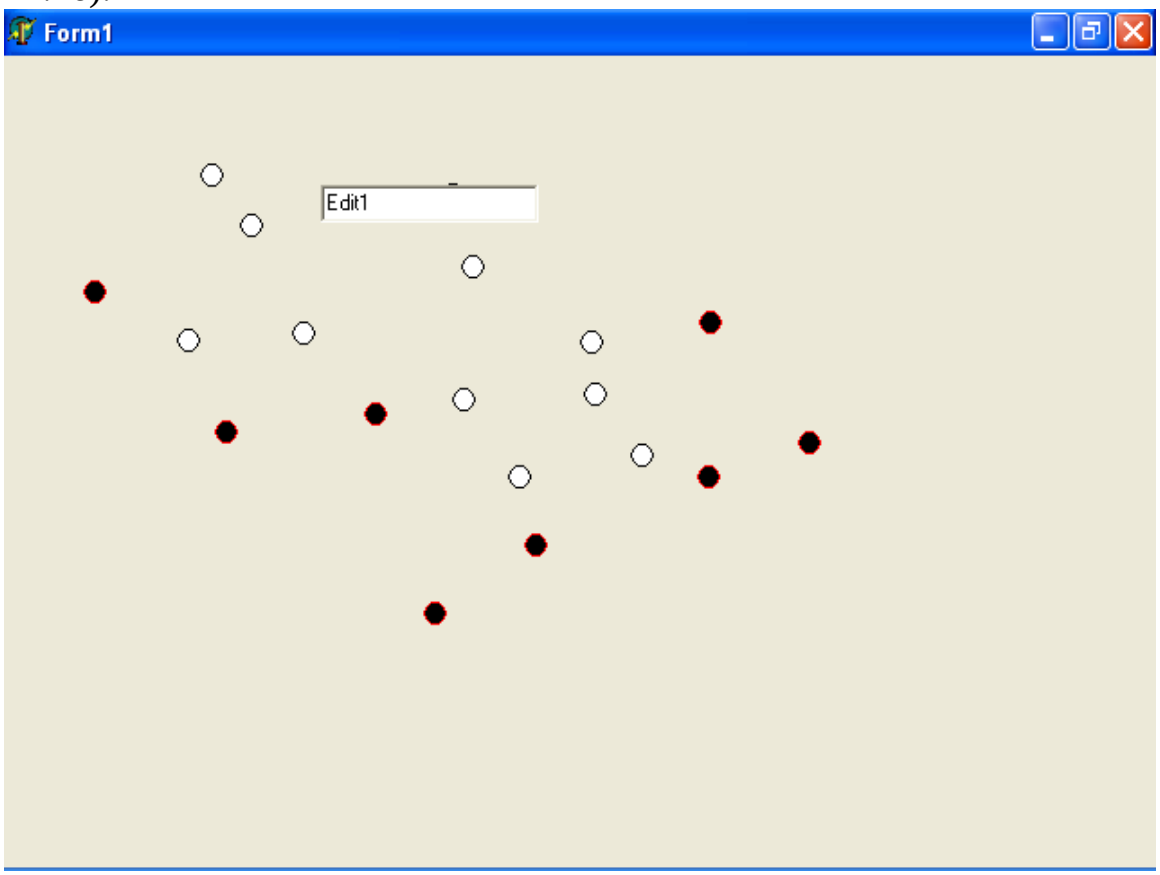


Рис. 1.16. Форма Form1 после обработки событий *OnMouseDoun*

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
  var NewShape:TShape;
begin
    NewShape:=  TShape.Create(Form1);
    NewShape.Parent:=Self;
    NewShape.Shape:=stCircle;
    NewShape.Height:=13;
    NewShape.Width:=13;
    NewShape.Left:=X;
    NewShape.Top:=Y;
    NewShape.show;
    if   Button=mbRight then
    begin
      NewShape.Pen.Color:=clRed;
      NewShape.Brush.Color:=clBlack;
    end;
end;
```

Задание. Поместите на форму компонент *TShape* в виде прямоугольника. В момент нажатия кнопки мыши на прямоугольник он исчезает. Когда кнопка будет отпущена, прямоугольник снова появится. Используйте процедуры *Hide, Show,OnMouseUp, OnMouseDown*. Добавьте в описание класса формы поле типа *MouseBtn: Boolean*. В процедуре обработки события *OnMouseMove* определить цвет компонента *TShape*: свойству *Color* присваивается в зависимости от значения поля *MouseBtn* значение *clAgua* (голубой) или *clYellow* (желтый).

Пример 2

В данном примере используется форма *Form*1, содержащая кнопку и группирующую рамку. После выполнения щелчка на кнопке она перемещается в группирующую рамку (рис. 1.17). Это обеспечивается путем задания свойству *Parent* кнопки значения *GroupBox*1. Двойной щелчок на *GroupBox*1 выводит *Button*1 за пределы группирующей рамки.

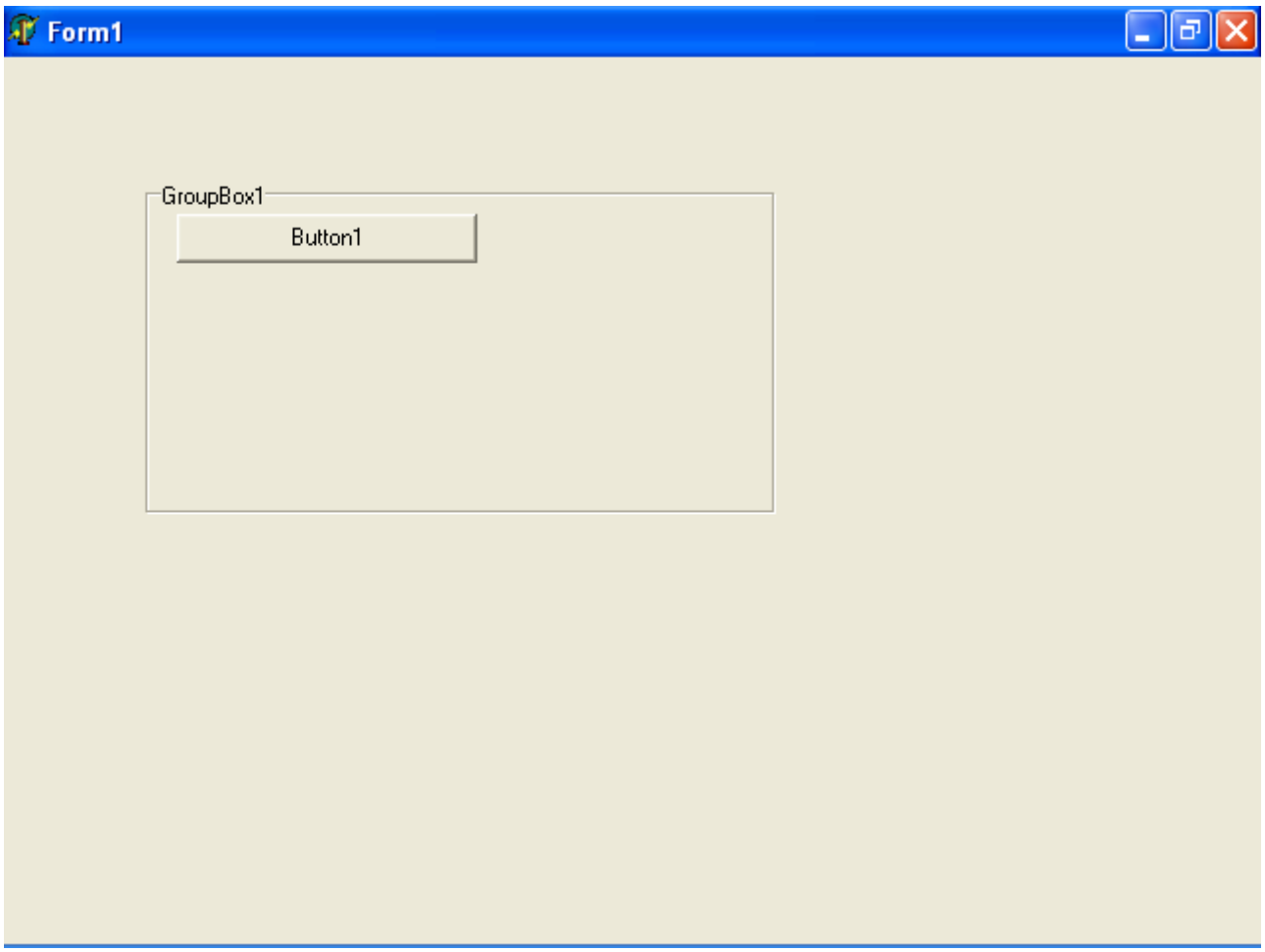


Рис. 1.17. Вид формы после нажатия на кнопку *Button*1

Ниже приведены соответствующие процедуры обработки событий:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Button1.Parent:=GroupBox1;
end;
procedure TForm1.GroupBox1DblClick(Sender: TObject);
begin
    Button1.Parent:=Form1;
end;
```

Задание. Добавьте в форму компонент *Memo*1. Покажите в нем имя родительского компонента. Используя компонент *DialogOpen*, вставьте в компонент *Memo* любой текстовый файл (лучше небольшой по размеру). Добавьте в форму кнопки *Button*2, *Button*3, *Button*4. Напишите процедуры выравнивания текста по правому, левому краю и в центре соответственно.

Пример 3

В форме находятся панель (*Panel*1), опция (*CheckBox*1), поле ввода (*Edit*1) и кнопка. Свойство *DragMode* имеет значение *DmAutomatic*. Когда пользователь выполняет щелчок на опции *CheckBox*1, начинается процесс ее перетаскивания. Когда перемещение компонента завершается, значение параметра *Target* в методе *CheckBox1EndDrag* определяет, какой текст появляется в окне сообщения. В поле ввода *Edit*1 отображается имя компонента, на котором была оставлена опция (форма после выполнения этих действий выглядит так, как показано на рис. 1.18):

```
procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
    Accept:=true;
end;
procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
    Accept:=true;
end;
```

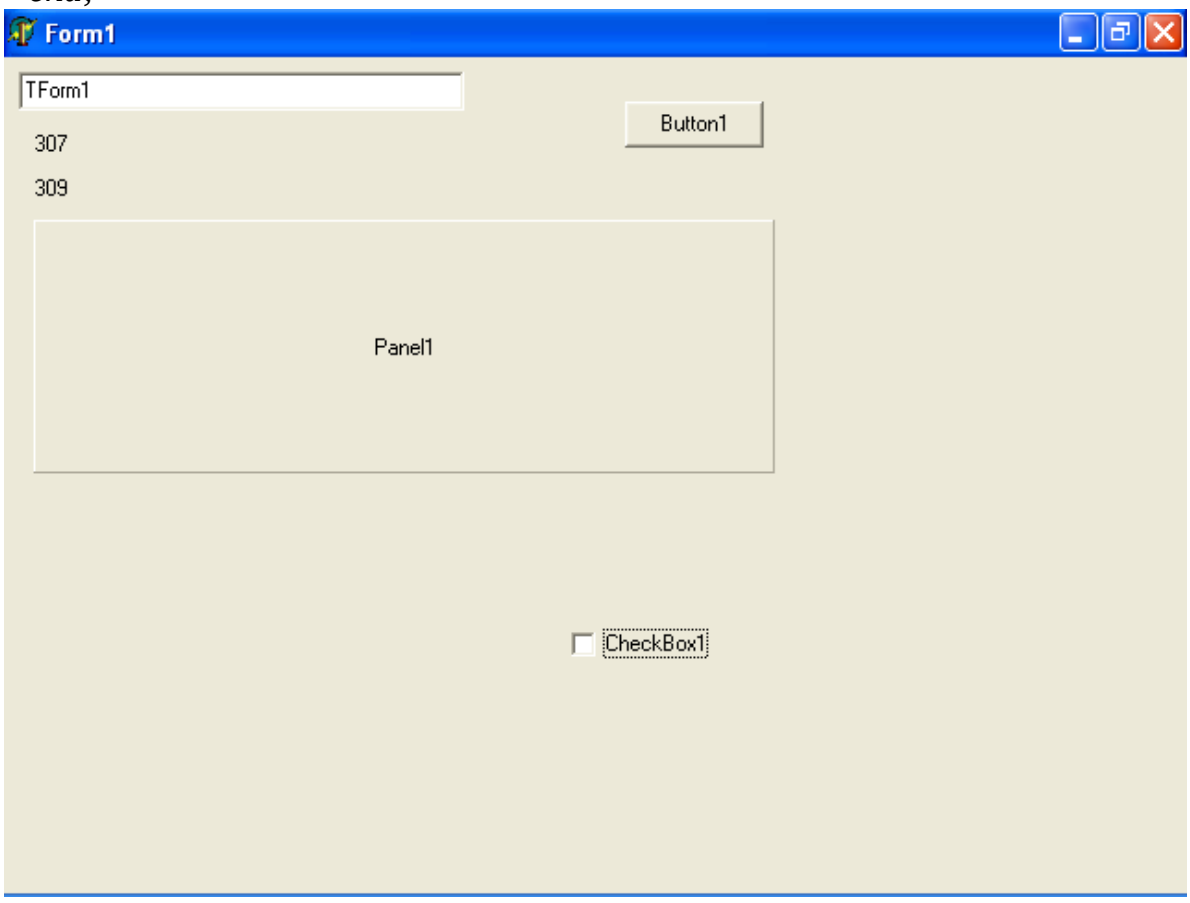


Рис. 1.18. Положение компонента *ChechBox*1 после перемещения на форму

```
procedure TForm1.CheckBox1EndDrag(Sender, Target: TObject; X, Y: Integer);
begin
    if Target<> nil  then
        Edit1.Text:= Target.ClassName
    else
        Edit1.Text:='нельзя перетащить объект в данном компоненте';
    if Target= Panel1  then
    begin
        showMessage('Оставлена на панели');
        CheckBox1.Left:=X+Panel1.Left;
        CheckBox1.Top:=Y+Panel1.top;
    end
    else begin
        if Target=Form1 then
        showMessage('Оставлена на Form1 ');
        CheckBox1.Left:=X;
        CheckBox1.Top:=Y;
    end;
       Label1.Caption:= IntTostr(X);
       Label2.Caption:= IntTostr(Y);
   CheckBox1.Left:=X;
   CheckBox1.Top:=Y;
end;
```

Задание. Пусть в форме содержится три опции. Свойство *DragMode* каждой из них имеет значение *dmAutomatic*. Используя функцию *Dragging*, определите, перетаскивается объект или нет. Когда опция перетаскивается, измените цвет формы на *clAgua*, *clYellow*, *clLime*, соответствующий каждой опции.

Пример 4

События клавиатуры выполняются процедурой обработки события того компонента, который в данный момент активен для ввода. Зададим клавишу, например [*F*6], с помощью которой пользователь может увеличить изображение на экране. Для компонента *Image*1 зададим его свойство *Picture* и поместим его в центр. Нажмем клавишу [*F*6] в компоненте*Edit*1 (см. рис. 1.19):

```
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if Key=VK_f6 then
     Edit1.Text:='F6 нажата';
    Image1.stretch:=true;
end;
```

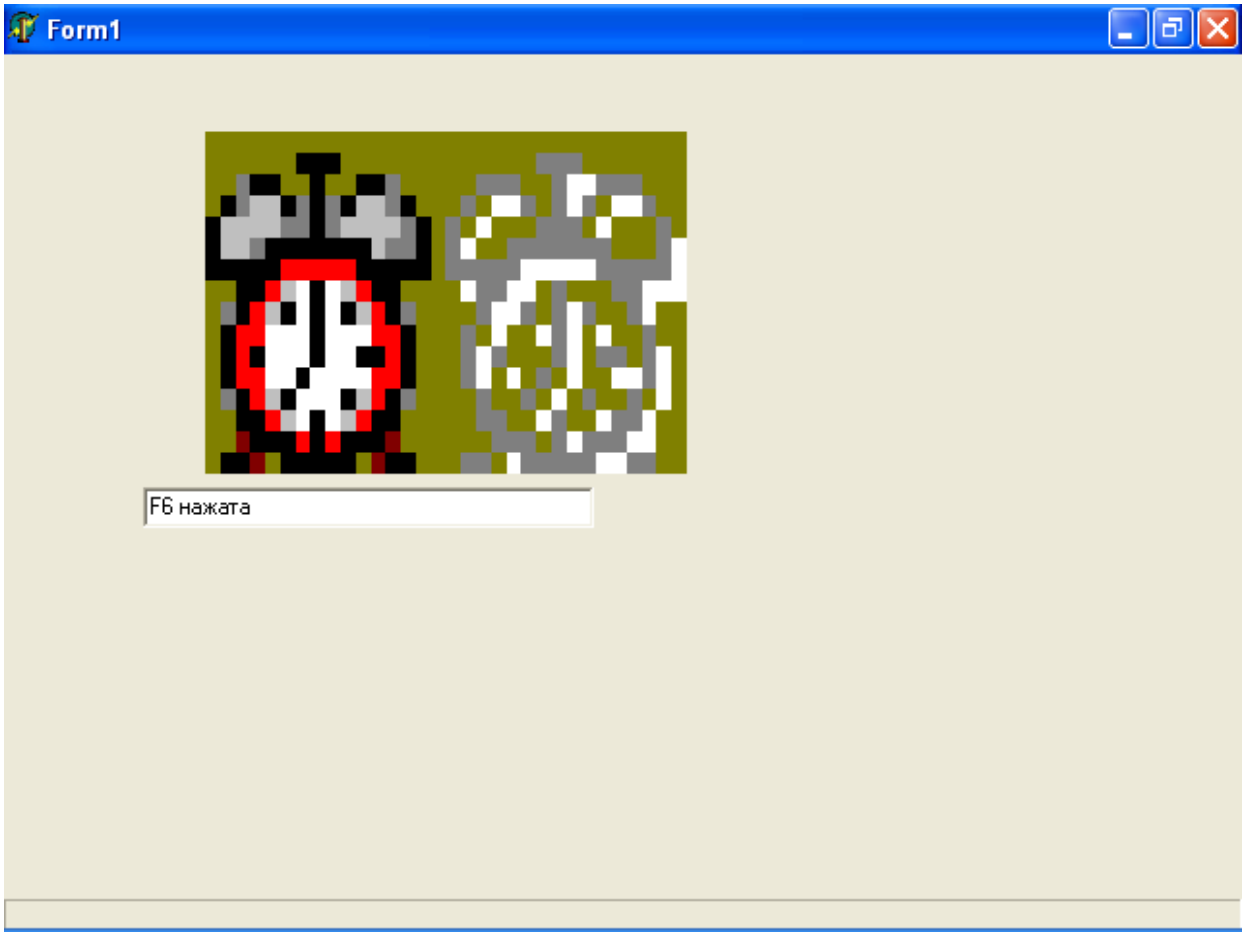


Рис. 1.19. Форма после нажатия клавиши *F*6

Задание. Установите свойство формы *KeyPreview true*. Определите, что изменилось. Выведите предупреждение пользователю, что цифровая клавиатура включена/не включена. Предупреждение выведите в компоненте *StatusBar*1 в *Panel*1, а сообщение из примера в *Panel*2.

Пример 5

Пример демонстрирует последовательность, в которой события клавиатуры обрабатываются приложением. Когда программа запускается в первый раз, при нажатии клавиш [*Ctrl+Alt*] форма приобретает фиолетовый цвет. Если выполнить щелчок на кнопке Button1, эта кнопка перестанет быть активной (см. рис. 1.20). При щелчке на *Button*2 свойство *KeyPreview* формы

приобретет значение *True*. Если теперь нажаты клавиши в сочетании [*Ctrl+Alt*], форма станет голубой, так как она обрабатывает события нажатия клавиш:

```
procedure TForm1.Button1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if (Shift=([ssAlt, ssCtrl])) then
    form1.Color:=clPurple;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
    Button1.Enabled:=false;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
    form1.KeyPreview:=True;
end;
```

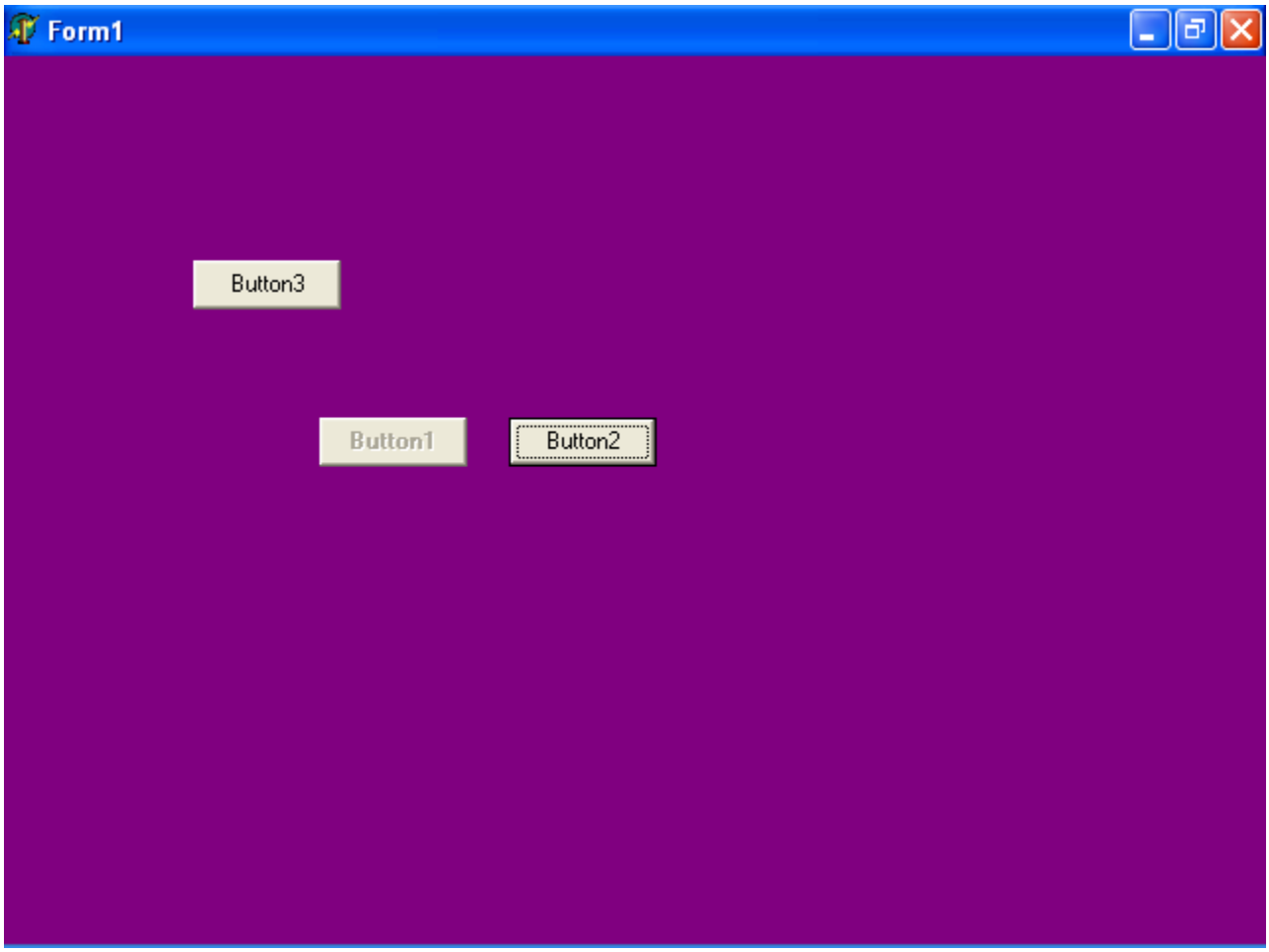


Рис. 1.20. Результат нажатия сочетания клавиш [*Ctrl+Alt*]

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if (Shift=([ssAlt, ssCtrl])) then
    form1.Color:=clAqua;
end;
```

Задание. Поменяйте элемент управления табулятором и нажмите сочетание клавиш [*Ctrl+Alt*]. Запустите эту форму еще раз. Добавьте еще одну кнопку *Button*3. Напишите обработчик события: кнопка *Button*3 активна, цвет кнопок *Button*1 и *Button*2 при сочетании клавиш [*Ctrl+F*6] меняется произвольным образом. Используйте функцию *Random*.

Пример 6

При нажатии клавиши обычно генерируется целый ряд событий. В примере используется список, отображающий, в какой последовательности обрабатываются нажатия клавиш с помощью обработчика событий *OnkeyDown*, *OnKeyUp* и *OnKeyPress*. Функция *ShortCutToText* объявлена в модуле *Menus*. Поскольку в данном разделе форма не имеет меню, то этот модуль не включается автоматически в разделе *Uses* модуля, поэтому вы должны добавить туда имя этого модуля самостоятельно. Поле *Edit*1 должно быть активным для ввода (см. рис. 1.21):

```
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    ListBox1.Items.Add('Edit1.KeyDown-'+ShortCutToText(Key));
end;
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    ListBox1.Items.Add('Edit1.KeyPress-'+Key);
end;
procedure TForm1.Edit1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    ListBox1.Items.Add('Edit1.KeyUp-'+ShortCutToText(Key));
end;
```

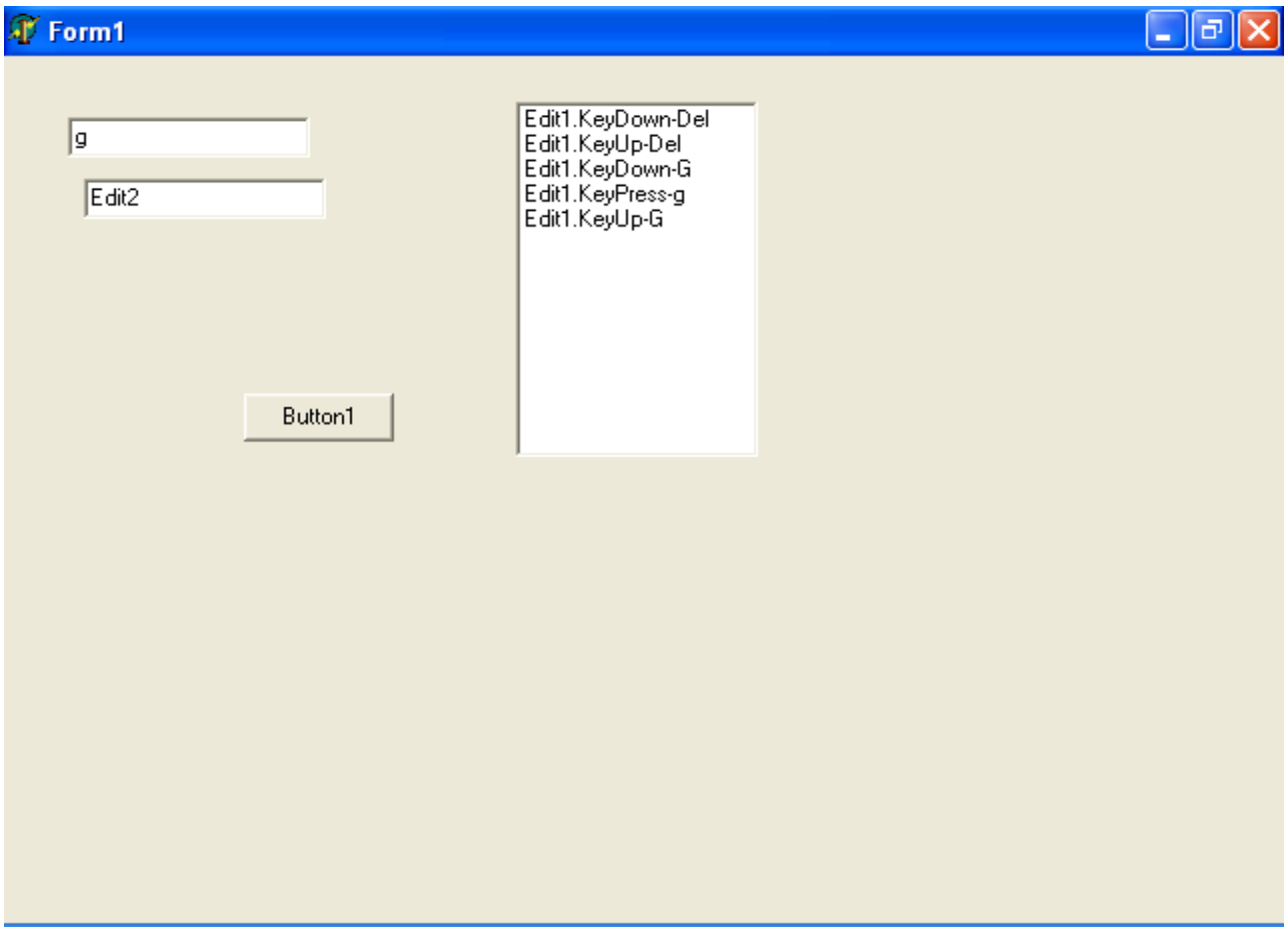


Рис. 1.21. Результат вызовов обработчиков событий *OnKeyDown*, *OnKeyPress, OnKeyUp*

Задание. Измените параметр функции *ShortCutToText*(*ShortCut: TShortCut*): *string* так, чтобы можно было увидеть сочетание клавиш. Используя функцию *TextToShortCut*(*Text: string*): *TshortCut*, покажите в каком-нибудь компоненте сочетание нажимаемых клавиш на компоненте *Edit*2.

Пример 7

Следующие процедуры обработки событий *FormKeyDown* и *FormKeyUp* включают и отключают изображения в компоненте *Image* и изменяют текст в компоненте панели *Panel*1, когда клавиша [*Z*] нажимается (см. рис. 1.22) и затем отпускается.

В этом примере параметр *Key* преобразуется в тип *Char*:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if chr(Key)='Z'  Then
    begin
        Image1.Visible:=true;
```

```
        panel2.Caption:=Загрузка судна в порту';
    end;
end;
procedure TForm1.FormKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
    if chr(Key)='Z'  Then
    begin
        Image1.Visible:=false;
        panel2.Caption:='';
    end;
end;
```

Рис. 1.22. Изображение и текст появятся, когда клавиша отжата

Задание. При нажатии клавиши [*Ins*] в компоненте панели *Panel*1 должен отобразиться текст. При нажатии сочетания клавиш *Alt*+*F*10 появится сообщение – информация: «нажаты *Alt*+*F*10». Используйте функцию:

function *MessageDlg(const Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint): Word.*

Пример 8

Форма содержит панель, меню и компонент *Bevel*. В свойствах *Hint* каждого компонента напишите справочный текст (в каждом пункте меню тоже). Процедура *OnHint* является обработчиком события компонента *Tapplication*, поэтому написана процедура обработки *OnHint*. Новый метод *DisplayHintPanel* включен в форму в процедуре *OnCreate* главной формы (результат работы программы показан на рис. 1.23):

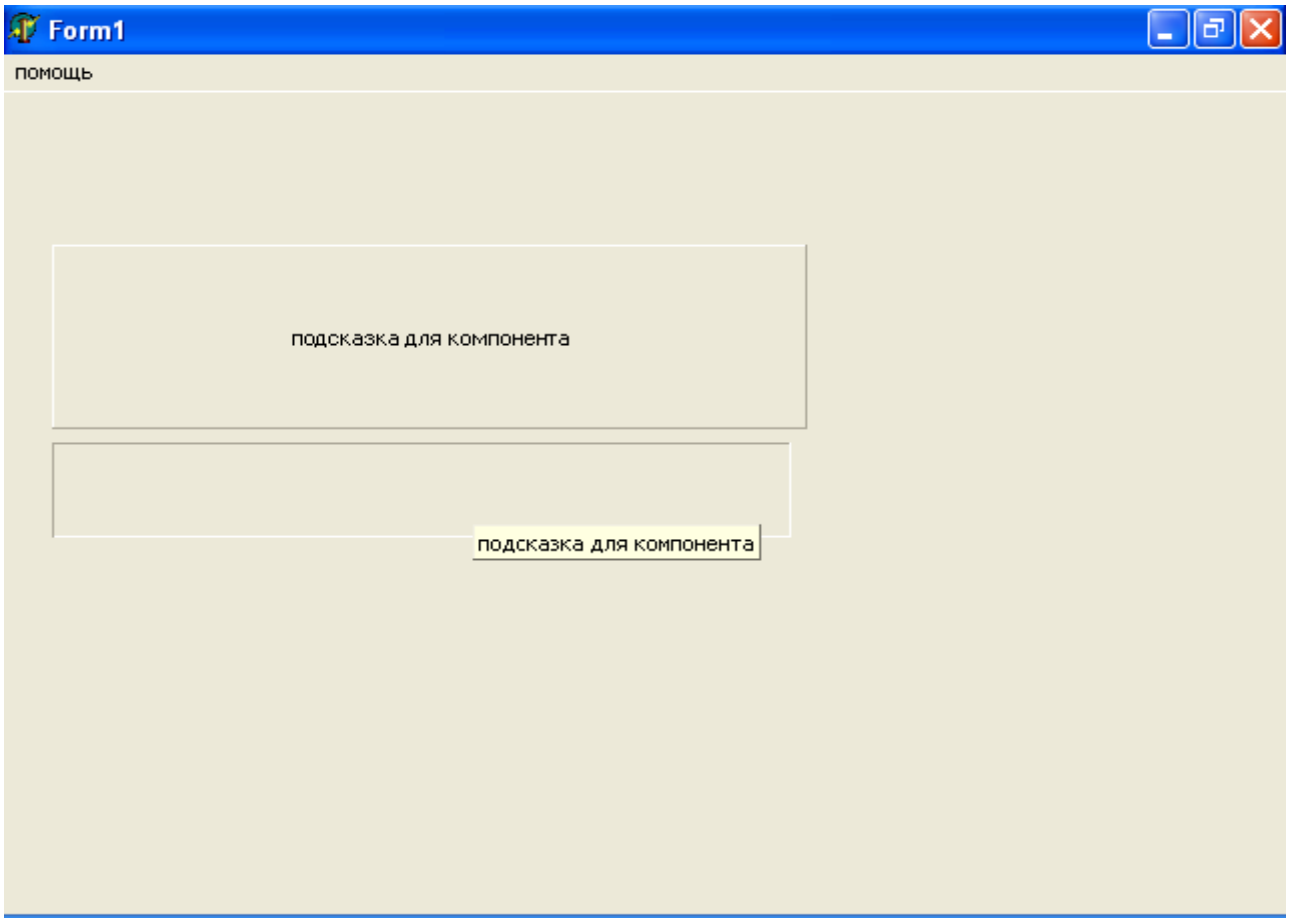


Рис. 1.23. Отображение подсказок для компонентов

```
unit Unit17;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
```

```
    MainMenu1: TMainMenu;
    Bevel1: TBevel;
    N2: TMenuItem;
    N1: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
  procedure DisplayHintPanel(Sender: TObject);
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
     Application.OnHint:=DisplayHintPanel;
end;
procedure TForm1.DisplayHintPanel(Sender: TObject);
begin
     Panel1.caption:= Application.hint;
end;
end.
```

Задание. В форму добавьте кнопку. По нажатию клавиши добавьте новый пункт меню. Подсказка содержит текст «подсказка к пункту» + название пункта. Текст названия пункта меню можно передать из компонента *Edit*.

Пример 9

В форме помещены компоненты: кнопка, *OpenPictureDialog1* и *Image*1. После открытия окна диалога в компонент *Image* вставляется выбранный файл с рисунком (пример работы программ показан на рис. 1.24):

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OpenPictureDialog1.Execute;
  Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;
```

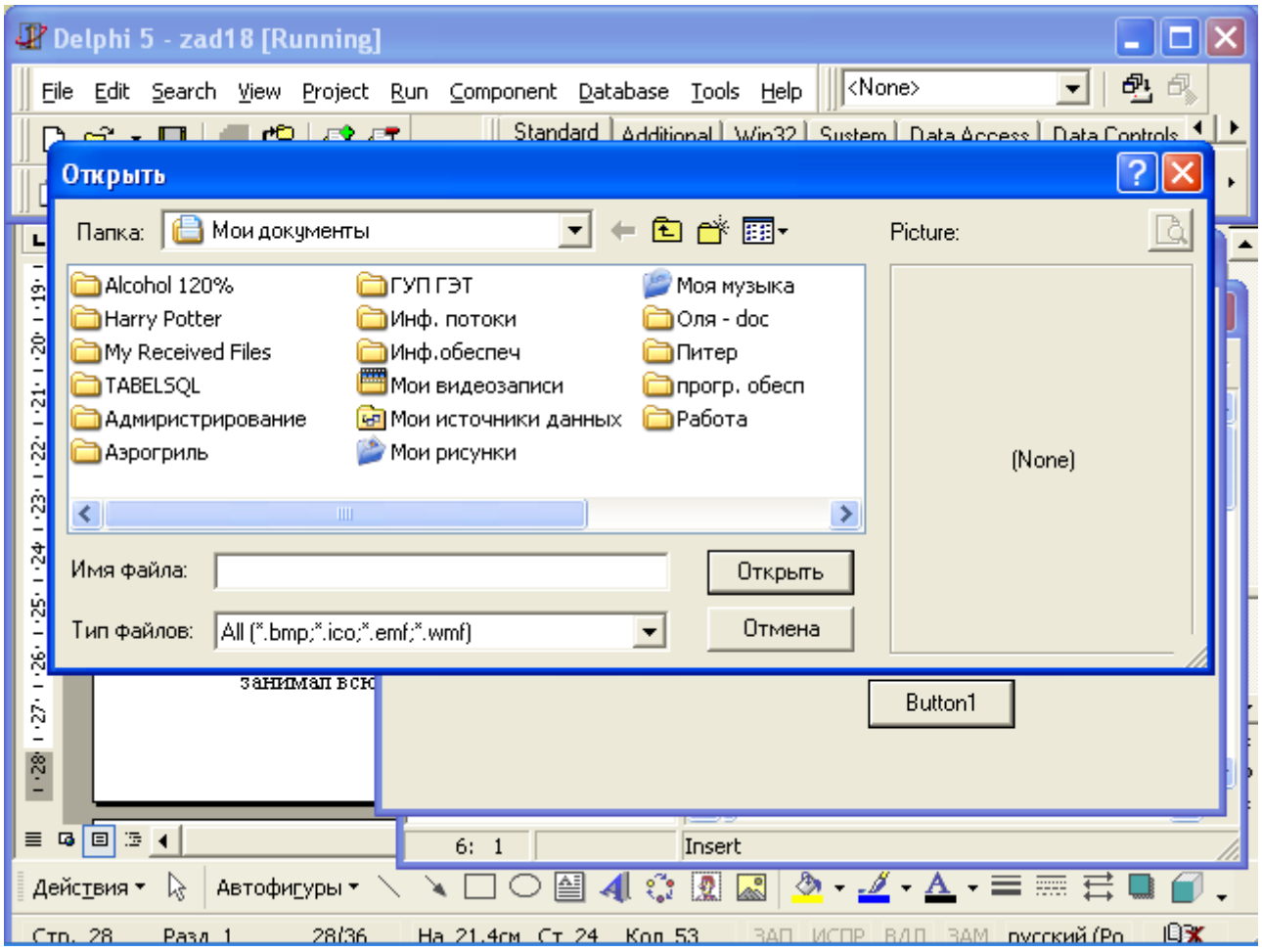


Рис. 1.24. Открытие компонента *OpenPictureDialog*1 для поиска рисунка

Задание. Настройте компонент OpenPictureDialog1 для открытия только рисунков с расширением **.bmp*. Настройте свойства *Image*1 так, чтобы рисунок занимал всю форму и входил полностью в компонент.

Пример 10

Для работы с документами используются приложения: *MDI* (*Multiple Document Interface*) – интерфейс для одновременной работы со многими документами и *SDI* (*Single Document Interface*) – интерфейс для работы с одним документом. В *MDI*-приложениях два или более окон могут быть активными. В *SDI*-приложениях это невозможно. Различные документы имеют общее рабочее пространство, называемое родительским окном. В *MDI*-приложении родительское окно всегда является главной формой приложения. В нем возможны другие окна, в которых могут быть открыты различные документы.

Эти окна не могут размещаться за пределами окна приложения. Они называются дочерними окнами.

В проект поместите четыре формы. В форме *Unit*1 в свойстве *FormStyle* установите значение *fsMDIForm* – родительскую форму. Все остальные формы определите как дочерние установкой значения *fsMDIChild*. Свойству *Visible* всех форм должно быть задано значение *True*. Включите в форму компонент *TMainMenu*. Назовите его *Window*. В это меню добавьте пункты *Cascade*, *Tile* и т.д. С их помощью можно управлять окнами *MDI*-приложения (рис. 1.25):

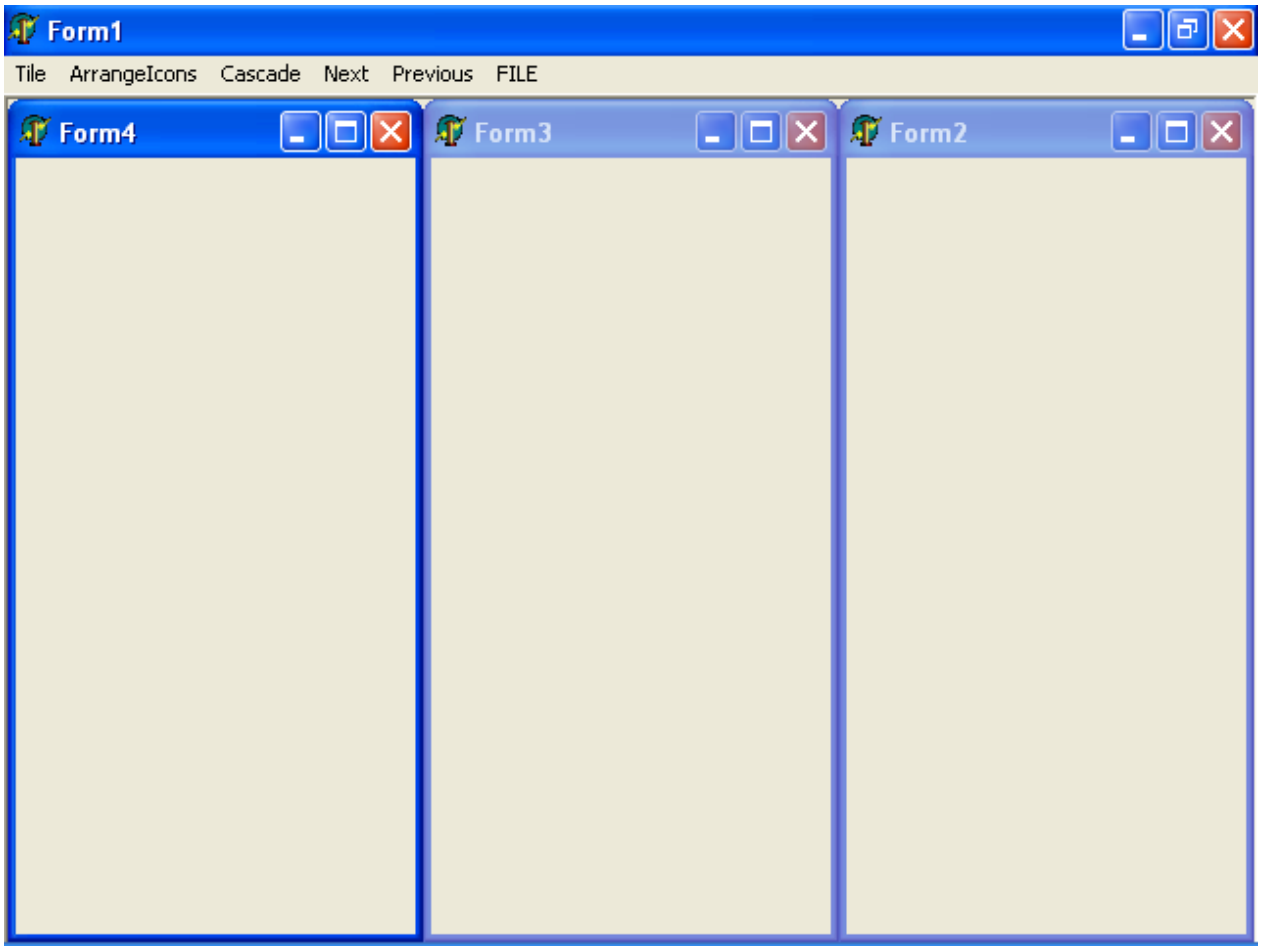


Рис. 1.25. Форма управления окнами

```
procedure TForm1.Tile1Click(Sender: TObject);
begin
    TileMode:=tbVertical;
    Tile;
end;
procedure TForm1.ArrangeIcons1Click(Sender: TObject);
begin
    ArrangeIcons;
end;
```

```
procedure TForm1.Cascade1Click(Sender: TObject);
begin
    Cascade;
end;
procedure TForm1.Next1Click(Sender: TObject);
begin
    Next;
end;
procedure TForm1.Previous1Click(Sender: TObject);
begin
    Previous;
end;
procedure TForm1.CLOSE1Click(Sender: TObject);
begin
    if ActiveMDIChild <> nil then
    activeMDIChild.close;
end;
end.
```

Задание. Каждой форме присвойте свой заголовок, используя свойство *Caption*. Для каждой дочерней формы установите разные значения свойства *WindowState*. На каждой дочерней форме расположите по кнопке *Button*1. На каждую кнопку напишите процедуру изменения свойств другой формы (*WindowState, Position, BorderStyle, Font*).

1.2.3. Глоссарий для среды Delphi

Active – свойство типа *Boolean* определяет активность формы, приложения, открытие или закрытие набора данных.

Acr – рисование дуги.

Add – метод, обеспечивающий добавление строк в список.

Align – свойство типа *Talign*, определяет вариант выравнивания компонента внутри контейнера.

Application – объект типа *TApplication* с именем *Application*, которое автоматически создается при каждом запуске приложения *Delphi*.

Clear – метод, служащий для очистки содержимого компонентов (текстовой информации).

Close – метод закрытия формы.

Create **–** метод создания экземпляров формы.

ComponentCount **–** число принадлежащих компонентов.

ComponentIndex **–** номер компонента в списке принадлежащих компонентов.

ComponentState **–** состояние текущего компонента.

Delete **–** метод удаления строк из списка.

Ellipse **–** рисование заполненного эллипса.

Execute **–** функция, используемая для вызова любого стандартного диалога. При закрытии диалога кнопкой *OK* (Открыть или Сохранить) функция *Execute* возвращает значение *True*, а при отмене диалога – значение *False*.

FillRect **–** заполнение прямоугольной области.

FrameRect **–** вывод незаполненного прямоугольника.

Form **–** компонент класса *Tform*; на основе формы начинается конструирование приложения.

FormStyle **–** свойство типа *TFormStyle* для определения стиля формы.

Free **–** метод уничтожения формы.

LineTo **–** рисование линии от указателя до точки с координатами *X* и *Y*.

Hide **–** метод управления видимостью. Процедура *Hide* скрывает форму, устанавливая ее свойству *Visible* значение *False*.

Hint **–** свойство типа *String*; задает текст подсказки, отображаемый в том случае, когда курсор находится в области компонента и некоторое время неподвижен.

Items **–** свойство типа *Tstring*; представляет собой массив строк и определят количество элементов списка и их содержимое.

KeyPreview **–** свойство типа *Boolean;* определяет, будет ли форма обрабатывать события клавиатуры прежде, чем их обрабатывают элементы управления формы.

MaxLength **–** свойство, определяющее, какое максимальное число символов можно ввести в поле ввода, элемент управления *Memo* или поле со списком.

Message **–** свойство типа *String*; содержит описание исключительной ситуации.

MessageDlg **–** функция, отображающая окно сообщений в центре экрана. Она позволяет получить ответ пользователя. Параметр *Msg* содержит выводимое сообщение.

MessageDlgPos **–** функция, отличающаяся от функции *MessageDlg* наличием параметров *X* и *Y*, управляющих положением окна на экране.

SelLength **–** свойство, возвращающее длину отмеченной в полк ввода подстроки (в символах).

SelectAll **–** метод, выделяющий в элементе весь текст.

SelStart **–** свойство, устанавливающее начало отмеченного диапазона.

Show **–** процедура, отображающая форму в немодальном режиме, при этом свойству *Visible* присваивается значение *True*. Сама форма переводится на передний план.

ShowMessage **–** процедура, отражающая окно сообщения с кнопкой *OK*. Заголовок содержит название исполняемого файла приложения, а строка *Msg* выводится как текст сообщения.

Insert **–** метод, задающий позицию в списке, на которую вставляется элемент.

InputBox **–** функция, которая отображает диалоговое окно, содержащее для ввода строку текста.

Tag **–** целое число, хранимое вместе с компонентом.

Name **–** имя компонента.

OnClick **–** событие нажатия, возникающее при выборе управляющего элемента.

OnKeyDown – событие, которое непрерывно генерируется при нажатии клавиши и удержании ее нажатой.

OnKeyPress **–** событие, которое генерируется при нажатии клавиши.

OnKeyUp **–** событие, которое происходит только после отпускания клавиши.

Owner **–** владелец компонента.

Parent **–** свойство типа *TwinControl*; указывает на родительский элемент управления для компонента.

Pie **–** вывод фигуры в форме сектора круга.

Polygon **–** рисование заполненного многоугольника.

PlyLine **–** рисование незаполненного многоугольника.

Rectagle **–** вывод заполненного прямоугольника.

ReadOnly **–** свойство, которое определяет, может ли пользователь изменить текст элементов управления.

RoundRect **–** вывод заполненного прямоугольника со скругленными краями.

Refresh **–** метод, используемый для обновления элементов управления, состоящего в удалении изображения элемента и его перерисовке.

Run **–**команда, которая компилирует, компонует и затем запускает на выполнение созданный *EXE*-файл приложения.

Text – свойство типа *Tcaption*; содержит строку, связанную с компонентом. Это содержимое компонента.

TabOrder – свойство типа *TtabOrder*; определяет порядок получения компонентами контейнера фокуса при нажатии клавиши <*Tab*>, т.е. последовательность обхода (табуляции) компонентов.

Value – свойство типа *Variant* – это физические данные в объекте.

Var – ключевое слово для объявления раздела переменных.

1.3. Разработка веб-приложений с помощью PHP

История *PHP* начинается с 1995 года. В 1997 году было решено, что сокращение *PHP* должно означать не «*Personal Home page*», а «*PHP Hypertext Processor*». *PHP* – язык программирования, предназначенный для создания web-страницы с использованием баз данных.

Одним из главных достоинств *PHP* является тот факт, что он внедряется прямо в *HTML*-код, поэтому программисту не приходится писать программу с множеством команд для простого вывода *HTML*. Код *HTML* и *PHP* можно чередовать по мере необходимости. *PHP* позволяет написать фрагмент следующего вида:

```
<html>
<title><? print "Hello world!"; ?></title>
</html>
```

1.3.1. Характеристика языка PHP

Главным фактором при проектировании языка PHP является практичность. *PHP* предоставляет программисту средства для быстрого и эффективного решения поставленных задач. Практический характер *PHP* обусловлен пятью важными характеристиками: традиционностью, простотой, эффективностью, безопасностью, гибкостью.

Существует еще одна «характеристика», которая делает *PHP* особенно привлекательным: он распространяется бесплатно.

Для работы с *PHP* необходимо загрузить, установить и настроить *PHP* и *WEB*-сервер на компьютере. *PHP* совместим с разными *WEB*-серверами. Лучше, что использовать *Apache* – во-первых, это самый популярный *WEB*-сервер на сегодняшний день, во-вторых, он чаще всего работает с *PHP*.

1.3.1.1. Переход в PHP

Механизм лексического анализа должен как-то отличать код *PHP* от других элементов страницы. Идентификация кода *PHP* называется «переходом

в *PHP*» (*escaping to PHP*). Существуют четыре варианта оформления перехода в *PHP*: стандартные теги; короткие теги; теги *script*; теги в стиле *ASP*.

1.3.1.2. Стандартные теги

Стандартные теги используются программистами *PHP* чаще остальных способов, что объясняется наглядностью и удобством этой формы записи (см. листинг 1):

Листинг 1. Вывод кода *HTML* средствами *PHP*

```
<?php print "Welcome to the world of PHP!": ?>
```

Весь текст, расположенный до закрывающего тега ?>, интерпретируется как код *PHP*.

Одной из самых замечательных особенностей *HTML* является простота использования в сочетании с другими языками — например, *HTML* и *JavaScript* (см. листинг 2).

Листинг 2. Вывод кода *HTML* средствами *PHP*

```
<html>
<head>
<title>Basic PHP/HTML integration</title>
</head>
<body>
<?
print "<h3>PHP/HTML integration is cool.</h3>";
?>
</body>
</html>
```

В листинге 3 продемонстрировано включение динамической информации в *WEB*-страницу на примере вывода текущей даты в заголовке окна.

Листинг 3. Динамический вывод даты

```
<title>PHP Recipes | <? print (date("F d, Y")); ?></title>
```

PHP также позволяет изменять формат конструкций HTML – для этого соответствующая характеристика тега присваивается переменной, вставляемой в файл. В листинге 4 эта возможность продемонстрирована на примере присваивания характеристики шрифта (*h*3) переменной *$big_font* и ее последующего использования при выводе текста.

Листинг 4. Динамические теги *HTML*

```
<html>
```

```
<head>
<title>PHP Recipes | <? print (date("F d, Y")); ?></title>
</head>
<?
$big_font = "h3";
?>
<body>
<? print "<$big_font>PHP Recipes</$big_font>"; ?>
</body>
</html>
```

1.3.1.3. Переменные и типы данных

Типы данных составляют основу любого языка программирования и являются средством, с помощью которого программист представляет разные типы информации. В *PHP* поддерживаются шесть основных типов данных: целые числа, вещественные числа, строки, массивы, объекты, логические величины.

Переменная представляет собой именованную область памяти, содержащую данные, с которыми можно выполнять операции во время выполнения программы.

Имена переменных всегда начинаются со знака доллара *$*. Ниже приведены примеры допустимых имен переменных:

$color, $operating_system, $_some_variable, $model.

Имена переменных должны соответствовать тем же условиям, что и идентификаторы. Другими словами, имя переменной начинается с буквы или символа подчеркивания и состоит из букв, символов подчеркивания, цифр или других *ASCII*-символов в интервале от 127 до 255.

Переменная объявляется при первом ее использовании в программе. Более того, тип переменной косвенно определяется по типу хранящихся в ней данных. Рассмотрим следующий пример:

$sentence = "This is a sentence."; // $sentence интерпретируется как строка;

$price = 42.99: // *$price* интерпретируется как вещественное число;

$weight = 185; // *$weight* интерпретируется как целое число.

Область видимости (*scope*) переменных определяется как область доступности переменной в той программе, в которой она была объявлена. В зависимости от области видимости переменные *PHP* делятся на четыре типа: локальные переменные; параметры функций; глобальные переменные; статические переменные.

1.3.1.4. Параметры функций

В *PHP* любые параметры, передаваемые функции при вызове, должны быть объявлены в заголовке функции. Хотя параметрам присваиваются аргументы, переданные извне, после выхода из функции они становятся недоступными.

Параметры объявляются в круглых скобках после имени функции. Объявление параметров практически не отличается от объявления типичной переменной: функция умножает переданное значение на 10 и возвращает результат (см. пример 1).

Пример 1

```
function x10 ($value)
{
  $value = $value * 10;
  return $value;
}
```

1.3.1.5. Глобальные переменные

Глобальные переменные, в отличие от локальных, доступны в любой точке программы. Но чтобы изменить значение глобальной переменной, необходимо специально объявить ее как глобальную в соответствующей функции. Для этого перед именем переменной ставится ключевое слово *GLOBAL* (см. пример 2).

Пример 2

```
$somevar = 15;
function addit()
{
  GLOBAL $somevar;
  $somevar++;
  print "Somevar is $somevar";
}
addit();
```

Будет выведено значение *$somevar*, равное 16.

1.3.1.6. Константы

Константой называется именованная величина, которая не изменяется в процессе выполнения программы. Константы особенно удобны при работе с заведомо постоянными величинами – например, числом π (3,141592). В *PHP*

константы определяются функцией *define*(). После того как константа будет определена, будет невозможно изменить (или переопределить) ее в этой программе (см. пример 3).

Пример 3

define("'PI", "3.141592");

1.3.1.7. Выражения

Выражение описывает некоторое действие, выполняемое в программе. Каждое выражение состоит, по крайней мере, из одного операнда и одного или нескольких операторов.

1.3.1.8. Операнды

Операнд представляет собой некоторую величину, обрабатываемую в программе. Примеры операндов:

$a++;

где *$a* – операнд;

$sum = $val1 + $val2;

где *$sum, $val1* и *$val2* – операнды.

1.3.1.9. Операторы и операции

Операция представляет собой символическое обозначение некоторого действия, выполняемого с операндами в выражении. Операция всегда возвращает некоторое значение (результат). Оператор управляет ходом выполнения программы. Примерами операторов являются операторы выбора и цикла.

В табл. 1.2 приведен полный список всех операций, упорядоченных по убыванию приоритета.

Таблица 1.2

Операции *PHP*

Операция	Ассоциативность	Назначение
( )	-	Изменение приоритета
new	-	Создание экземпляров объектов
! ~	П	Логическое отрицание, поразрядное отрицание
++ --	П	Инкремент, декремент
@	П	Маскировка ошибок
/ * %	Л	Деление, умножение, остаток

Окончание табл. 1.2

Операция	Ассоциативность	Назначение
+ - .	Л	Сложение, вычитание, конкатенация
<< >>	Л	Сдвиг влево, сдвиг вправо (поразрядный)
< <= > >=	–	Меньше, меньше или равно, больше, больше или равно
== != === <>	–	Равно, не равно, идентично, не равно
& ^ |	Л	Поразрядные операции *AND, XOR* и *OR*
&& ||	Л	Логические операции *AND* и *OR*
?:	П	Тернарный оператор
= += *= /= .= %= &= |= ^= <<= >>=	П	Операторы присваивания
AND XOR OR	Л	Логические операции *AND, XOR* и *OR*

Примеры выражений:

$a = 5; – присвоить целое число 5 переменной *$a*;

$a = "5";* – присвоить строковую величину "5" переменной *$a.

1.3.1.9.1. Математические операции

Математические операции (табл. 1.3) предназначены для выполнения различных математических действий.

Таблица 1.3

Математические операции

Пример	Название	Результат
$a + $b	Сложение	Сумма *$a* и *$b*
$a-$b	Вычитание	Разность *$a* и *$b*
$a*$b	Умножение	Произведение *$a* и *$b*
$a/$b	Деление	Частное от деления *$a* на *$b*
$a % $b	Остаток	Остаток от деления *$a* на *$b*

1.3.1.9.2. Операции присваивания

Операции присваивания задают новое значение переменной (табл. 1.4).

Таблица 1.4

Операции присваивания

Пример	Название	Результат
$a = 5;	Присваивание	Переменная *$a* равна 5
$a += 5;	Сложение с присваиванием	Переменная *$a* равна сумме *$a* и 5
*$a *= 5;*	Умножение с присваиванием	Переменная *$a* равна произведению *$a* и 5
$a/=5;	Деление с присваиванием	Переменная *$a* равна частному от деления *$a* на 5
$a .= 5;	Конкатенация с присваиванием	Переменная *$a* равна конкатенации $a и 5

1.3.1.9.3. Логические операции

Логические операции (табл. 1.5) обеспечивают средства для принятия решений в зависимости от значения переменных. Логические операции позволяют управлять порядком выполнения команд в программе и часто используются в управляющих конструкциях (таких, как условная команда *if*, а также циклы *for* и *while*).

Таблица 1.5

Логические операции

Пример	Название	Результат
$a && $b	Конъюнкция	Истина, если истинны оба операнда
$a and $b	Конъюнкция	Истина, если истинны оба операнда
$a OR $b	Дизъюнкция	Истина, если истинен хотя бы один из операндов
!$a	Отрицание	Истина, если значение *$a* ложно
NOT !$a	Отрицание	Истина, если значение *$a* ложно
$a XOR $b	Исключающая дизъюнкция	Истина, если истинен только один из операндов

Логические операции часто используются для проверки результата вызова функций:

file_exists("filename.txt") OR print "File does not exist!".

1.3.1.9.4. Операции равенства

Операции равенства (табл. 1.6) предназначены для сравнения двух величин и проверки их эквивалентности.

Таблица 1.6

Операции равенства

Пример	Название	Результат
$a==$b	Проверка равенства	Истина, если *$a* и *$b* равны
$a! = $b	Проверка неравенства	Истина, если *$a* и *$b* не равны
$a===$b	Проверка идентичности	Истина, если *$a* и *$b* равны и имеют одинаковый тип

1.3.1.9.5. Операции сравнения

Операции сравнения (табл. 1.7), как и логические операции, позволяют управлять логикой программы и принимать решения при сравнении двух и более переменных.

Операции сравнения предназначены для работы только с числовыми значениями.

Таблица 1.7

Операции сравнения

Пример	Название	Результат
$a<$b	Меньше	Истина, если переменная *$a* меньше *$b*
$a>$b	Больше	Истина, если переменная *$a* больше *$b*
$a<= $b	Меньше или равно	Истина, если переменная *$a* меньше или равна *$b*
$a>= $b	Больше или равно	Истина, если переменная *$a* больше или равна *$b*
($a-12*)*?5: -1	Тернарная операция	Если переменная *$a* равна 12, возвращается значение 5, а если не равна, то возвращается 1

1.3.1.9.6. Операторы выбора – if...else

К операторам выбора относят: условный оператор *(if...else)* и переключатель *(switch)*. Синтаксис условного оператора: *if(condition) statement* 1 *else statement* 2.

Условие *condition* может быть любым выражением. Если оно истинно, то выполняется оператор *statement* 1. В противном случае выполняется оператор *statement* 2.

1.3.1.9.7. Операторы выбора – переключатель switch

Переключатель *switch* является наиболее удобным средством для организации мультиветвления. Синтаксис переключателя таков:

```
switch(expression) – переключающее выражение
{
  case value1: – константное выражение 1
    statements; – блок операторов
  break;
  case value2: – константное выражение 2
    statements;
  break;
  default:
    statements;
}
```

1.3.1.9.8. Операторы цикла – while

Оператор *while* называется оператором цикла с предусловием. При входе в цикл вычисляется выражение условия, и, если его значение отлично от нуля, выполняется тело цикла. Затем вычисления выражения условия и операторов тела цикла выполняются до тех пор, пока значение выражения условия не станет равным нулю (пример 4).

Пример 4

```
<?
  $var = 5;
  $i = 0;
  while(++$i <= $var)
  {
    echo($i); echo('<br>');
  }
?>
```

1.3.1.9.9. Операторы цикла – do…while

Оператор цикла – *do...while* называется оператором цикла с постусловием. При входе в цикл в любом случае выполняется тело цикла (т.е. цикл всегда будет выполнен хотя бы один раз), затем вычисляется условие, и, если оно не равно 0, вновь выполняется тело цикла (см. пример 5).

Пример 5

```
<?
  $var = 5;
  $i = 0;
  do
  {
    echo($i); echo('<br>');
  }
  while(++$i <= $var)
?>
```

Итерационный цикл имеет следующий формат:

```
  for(expression1;expression2;expression3)
  {
    statements;
  }
```

Здесь *expression1* (инициализация цикла) – последовательность определений и выражений, разделяемая запятыми. Все выражения, входящие в инициализацию, вычисляются только один раз при входе в цикл. Как правило, здесь устанавливаются начальные значения счетчиков и параметров цикла (пример 6).

Пример 6

```
<?
  $var = 5;
  $i = 0;
  for ($i = 0; $i <= $var; $i++)
  {
    echo($i);
    echo('<br>');
  }
?>
```

1.3.1.10. Определение и вызов функций

Функции могут создаваться в любой точке программ *PHP*, однако по соображениям структурной организации кода удобнее разместить все функции,

используемые сценарием, в самом начале сценарного файла. Определение функции обычно состоит из трех частей:

- имени функции;
- круглых скобок, в которых перечисляются необязательные входные параметры, разделенные запятыми;
- тела функции, заключенного в фигурные скобки.

Обобщенный синтаксис функций *PHP* выглядит так:

function имя_функции ([$параметр1, $параметр2, ... $параметрN])

{

тело функции

}

1.3.1.11. Создание массивов

Массив представляет собой совокупность объектов, имеющих одинаковый размер и тип. Каждый объект в массиве называется элементом массива. При объявлении индексируемого массива после имени переменной ставится пара квадратных скобок ([ ]):

$languages[ ] = "Spanish".

После этого в массив можно добавлять новые элементы, как показано ниже. Обратите внимание: новые элементы добавляются без явного указания индекса. В этом случае новый элемент добавляется в позицию, равную длине массива плюс 1:

$languages[ ] = "English";

$languages[ ] = "Gaelic".

Функция *array*( ) получает ноль или более элементов и возвращает массив, состоящий из указанных элементов. Ее синтаксис:

array array([элемент1, элемент2...]).

Ниже показан пример использования *array*( ) для создания индексируемого массива:

$languages = array ("English". "Gaelic". "Spanish").

1.3.1.11.1. Многомерные массивы

Многомерный массив (массив массивов) – средство для хранения информации, требующее дополнительного структурирования. Создать многомерный массив несложно – просто добавьте дополнительную пару квадратных скобок, чтобы вывести массив в новое измерение:

$chessboard[1][4] = "King" – двухмерный массив;

$capitals["USA"]["Ohio"] = "Columbus" – двухмерный массив;

$streets["USA"]["Ohio"]["Columbus"] = "Harrison" – трехмерный массив.

1.3.1.11.2. Сортировка массивов

В *PHP* существуют стандартные функции сортировки (табл. 1.8).

Таблица 1.8

Функции сортировки

Функция	Сортировка	Обратный порядок	Сохранение пар «ключ/значение»
sort	Значение	Нет	Нет
rsort	Значение	Да	Нет
asort	Значение	Нет	Да
arsort	Значение	Да	Да
ksort	Ключ	Нет	Да
krsort	Ключ	Да	Да
usort	Значение	?	Нет
uasort	Значение	?	Да
uksort	Ключ	?	Да

1.3.1.12. Простые ссылки

По ссылкам пользователь может переходить как на обычные страницы *HTML*, так и на страницы, содержащие код P*H*P:

<View today's date.

Если щелкнуть на ссылке, в браузере будет загружена страница с именем *date.php*.

1.3.2. Основные структуры HTML документа

1.3.2.1. Структура документа HTML

Документы в языке *HTML* начинаются с декларации <!*DOCTYPE*>, затем следует элемент *HTML*, внутри которого содержатся последовательно элементы *HEAD* и *BODY* (см. листинг 4):

Листинг 4. Структура документа *HTML*

```
<HTML>
<HEAD>
  <TITLE> Текст заголовка</TITLE>
```

</HEAD>

<BODY>

тело документа

</BODY>

</HTML>

Минимальный документ *HTML* выглядит следующим образом (см. листинг 5):

Листинг 5. Пример *hello.html*

<TITLE>Hello</TITLE>

Результат листинга 5: *Hello world*.

1.3.2.2. Элемент HEAD и его производная TITLE

<!ELEMENT TITLE - - (#PCDATA) -(%head.misc)>*

Согласно спецификации *HTML*, каждый документ обязан иметь ровно один элемент *TITLE* в поле *HEAD*. С его помощью программе конечного пользователя сообщается название-уведомление данного документа, которое может быть выставлено в заголовке над окном соответствующей программы и т.д. Пример элемента *TITLE*:

<TITLE>Изучение динамики популяции</TITLE>.

1.3.2.3. Элемент BODY и его производные

Данный элемент содержит собственно тело (текст) документа. В теле документа может содержаться достаточно большой набор элементов: заголовки (*H*1 – *H*6), элемент *ADDRESS*, блочные элементы, элементы на уровне текста.

1.3.2.3.1. Заголовки

<!ELEMENT (%heading) - - (%text;)>*

<!ATTLIST (%heading) align (left|center|right) #IMPLIED >

Элементы *H*1, *H*2, *H*3, *H*4, *H*5 и *H*6 используются в документе для разметки заголовков. Всегда нужно указывать как начальный, так и конечный тэги. При этом заголовки, размеченные элементами *H*1, главенствуют над заголовками, размеченными элементами *H*2.

1.3.2.3.2. Блочные элементы разметки

P – параграфы. Для этого элемента разметки параграфов необходимо указывать начальный тэг, однако при этом конечный тэг может быть всегда опущен

UL – неупорядоченные списки. В данном случае обязательно указывать как начальный, так и конечный тэги, а также один или несколько элементов *LI*,

представляющие отдельные пункты списка. В элементах *UL* и *LI* может использоваться атрибут *TYPE*, устанавливающий стиль разметки для данного списка. Неупорядоченные списки имеют вид:

<UL>

<LI> ... первый пункт списка

<LI> ... второй пункт списка

...

</UL>.

OL – упорядоченные (т.е. нумерованные) списки. Здесь также требуется указывать как начальный, так и конечный тэги, а также один или несколько элементов *LI*, представляющие в списке отдельные пункты. Упорядоченные (или нумерованные) списки имеют следующий вид:

<OL>

<LI> ... первый пункт списка

<LI> ... второй пункт списка

...

</OL>.

DL – списки определений. Требуется указывать начальный и конечный тэги. В таком списке элементом *DT* размечается термин, а элементом *DD* – соответствующее ему определение. Списки определений имеют вид:

<DL>

<DT> название термина

<DD> определение термина

...

</DL>.

Заметим, что элементы *DT* можно использовать только как контейнеры для элементов текстового уровня. Но в то же время внутри *DD* можно использовать блочные элементы (исключение составляют заголовки и элементы *address*). Пример списка определений:

<DL>

<DT>Первый термин<dd>Определение первого термина.

<DT>Второй термин<dd>Определение для второго термина.

</DL>.

В результате должна получиться следующая разметка:

Первый термин

Определение первого термина.

Второй термин

Определение для второго термина.

PRE используется, когда необходимо включить в текст документа уже отформатированный текст. Для данного элемента необходимо указывать начальный и конечный тэги. Внутри такого элемента текст печатается шрифтом фиксированной ширины и при этом сохраняется разметка оригинала (пробелы и символы конца строк). Пример использования элемента *PRE*:

<PRE>

Higher still and higher

From the earth thou springest

Like a cloud of fire;

The blue deep thou wingest,

And singing still dost soar, and soaring ever singest.

</PRE>

будет воспроизведено как:

Higher still and higher

From the earth thou springest

Like a cloud of fire;

The blue deep thou wingest,

And singing still dost soar, and soaring ever singest.

DIV – деление документа на отдельные блоки. Для данного элемента необходимо указывать как начальный, так и конечный тэги.

CENTER – выравнивание текста. Для этого элемента необходимо указывать начальный и конечный тэги. Используется для центрирования стоящего между ними текста.

FORM – заполняемые формы. Необходимо указывать и начальный, и конечный тэги. Используется для создания заполняемой формы, которая будет обрабатываться на HTTP сервере.

HR – горизонтальные линейки. Не является контейнером, так что использовать конечный тэг нельзя.

TABLE – таблица, может быть вложенной.

1.3.2.3.3. Таблицы

В общем случае таблицы имеют следующий вид:

<TABLE BORDER=3 CELLSPACING=2 CELLPADDING=2 WIDTH="80%">

<CAPTION> ... заголовок таблицы ... </CAPTION>

<TR><TD> первая клетка таблицы <TD> вторая клетка

<TR> ...

</TABLE>

Для элемента *TABLE* всегда необходимо указывать как начальный, так и конечный тэги. При этом разрешается использовать следующие атрибуты:

align – данный атрибут принимает одно из следующих значений: *LEFT*, *CENTER* или *RIGHT* (используемый при этом регистр значения не имеет). Указывает для текущей таблицы, каким образом при разметке осуществляется ее горизонтальное выравнивание;

width – в отсутствие данного атрибута ширина таблицы определяется автоматически;

border – этот атрибут позволяет задавать для таблицы ширину внешней рамки – в пикселах (например, *BORDER* = 4). Данному атрибуту может быть также присвоено значение нуль, чтобы полностью отказаться от внешней рамки;

cellspacing – в языке *HTML* каждая ячейка имеет собственную границу, отделенную промежутком от границ соседних ячеек;

cellpadding – данный атрибут устанавливает для каждой ячейки в таблице расстояние в пикселах между рамкой ячейки и содержащимся в ней материалом.

Элемент *CAPTION* может иметь только один атрибут – *ALIGN*, который может принимать два значения: *ALIGN = TOP* или *ALIGN = BOTTOM*.

Для *TR*–элемента, начинающего новый ряд таблицы, необходимо указывать начальный тэг, однако всегда можно опустить конечный тэг. Элемент *TR* выступает в роли контейнера для ячеек таблицы и может иметь два атрибута:

align – устанавливает стиль горизонтального выравнивания для содержимого ячейки, который будет использоваться по умолчанию. Атрибут может принимать одно из следующих значений (независимо от используемого регистра): *LEFT*, *CENTER* или *RIGHT* и выполняет ту же самую роль, что и атрибут *ALIGN* при разметке параграфов;

valign – данный атрибут может использоваться при выборе правила, согласно которому – если нет других указаний – будет осуществляться вертикальное выравнивание во всех ячейках данной строки. Атрибут может принимать одно из следующих значений (независимо от используемого регистра): *TOP*, *BOTTOM* или *MIDDLE*. При этом содержимое ячейки будет,

соответственно, выравниваться по ее верхнему или нижнему краю, либо посередине.

1.3.2.4. Элементы текстового уровня

1.3.2.4.1. Элементы, задающие шрифт, используемый при разметке документа

Для всех элементов данного класса требуется указывать как начальный, так и конечный тэг, например:

<B>жирный текст</B>.

<B>жирный и <I>наклонный текст</I></B>.

Набор шрифтов, оговариваемых в спецификации *HTML*, ограничивается разметкой речи, либо должен применяться в документе для осуществления различных типов выделения:

TT – телетайпный текст или текст фиксированной ширины;

I – стиль с наклонным шрифтом;

B – стиль с жирным шрифтом;

U – стиль с подчеркиванием текста;

STRIKE – стиль с перечеркиванием текста;

BIG – печать текста шрифтом увеличенного размера;

SMALL – печать текста шрифтом уменьшенного размера;

SUB – печать текста со сдвигом вниз (нижний индекс);

SUP – печать текста со сдвигом вверх (верхний индекс).

1.3.2.4.2. Элементы разметки фраз

В спецификации *HTML* документа применяют следующие элементы разметки фраз:

EM – основной элемент, используемый в HTML для выделения текстов, обычно реализуется с привлечением наклонного шрифта;

STRONG – усиленное выделение, обычно реализуемое посредством жирного шрифта;

DFN – дает пример для обсуждаемого термина (понятия);

CODE – используется для выделения программного кода;

SAMP – используется при предоставлении в документе распечаток программ, программных сценариев и т.д.;

KBD – используется для выделения текста, который пользователь должен набрать на клавиатуре;

VAR – используется для обозначения переменных, либо аргументов, передаваемых с командами;

CITE – используется для обозначения цитат или ссылок на другие источники.

1.3.3. FORM (форма) – заполняемая форма

1.3.3.1. Атрибуты формы

Форма используется для таких действий пользователя, как регистрация, упорядочение пользователя или формирование запроса. Основной синтаксис формы и ее возможные атрибуты представлены в табл. 1.9:

<FORM ACTION="URL">

Таблица 1.9

Возможные атрибуты формы

Имя атрибута	Возможные значения	Смысл атрибута	Примечания
ACTION	*URL*	адрес сервера, который использует форма	сервер *HTTP* или *URL*
METHOD	*GET, POST*	метод передачи данных, полученных от пользователя, на сервер	по умолчанию – *GET*
ENCTYPE	строка	механизм, используемый для кодирования содержимого формы	по умолчанию приложение */x-www-form-url-кодирование*

Есть некоторые элементы, которые могут появиться только в пределах элемента *FORM*. В частности:

INPUT – текстовое однострокoвое поле, поля пароля, переключатели, радиокнопки, кнопки установки и перезагрузки, скрытые поля, кнопки выгрузки файла, кнопки изображений и т.д.;

SELECT – меню единичного или множественного выбора;

TEXTAREA – многострочное текстовое поле.

1.3.3.2. Способы обработки данных

PHP создает следующие массивы при обработке данных, пришедших из формы:

$_*GET* – содержит *GET*-параметры, переданные к скрипту через переменную окружения *QUERY_STRING*. Например, *$_GET('login')*;

$_*POST* – данные формы, полученные с помощью метода *POST*;

$_*COOKIE* – все *cookies*, которые прислал браузер;

$_*REQUEST* – объединение всех перечисленных выше массивов. Именно эту переменную рекомендуется использовать в скриптах, потому что таким образом мы не «привязываемся» жестко к типу принимаемых данных (*GET* или *POST*);

$_*SERVER* – содержит переменные окружения, переданные сервером.

При отладке сценария можно использовать переменную $*GLOBALS*, например (см. листинг 6):

Листинг 6. Файл отладки.

```
<!-- Выводит все глобальные переменные -->
<pre>
<? Print_r ($GLOBALS);
</pre>
```

1.3.3.3. Передача данных командной строкой

Пусть на сервере в корневом каталоге есть сценарий *PHP* под названием *test_forma.php* (см. листинг 7, рис. 1.26). Сценарий распознает два параметра: *login* и *password*:

Листинг 7. Страница с формой *test_forma.php*.

```
<html><body>
<?php
echo '<form action=test_pw.php>
Логин: <input type=text name="login" value="root"><br>
Пароль: <input type=password name="password" value="zz"><br>
<input type=submit value="Нажмите кнопку">
</form>';
?>
</body></html>
```

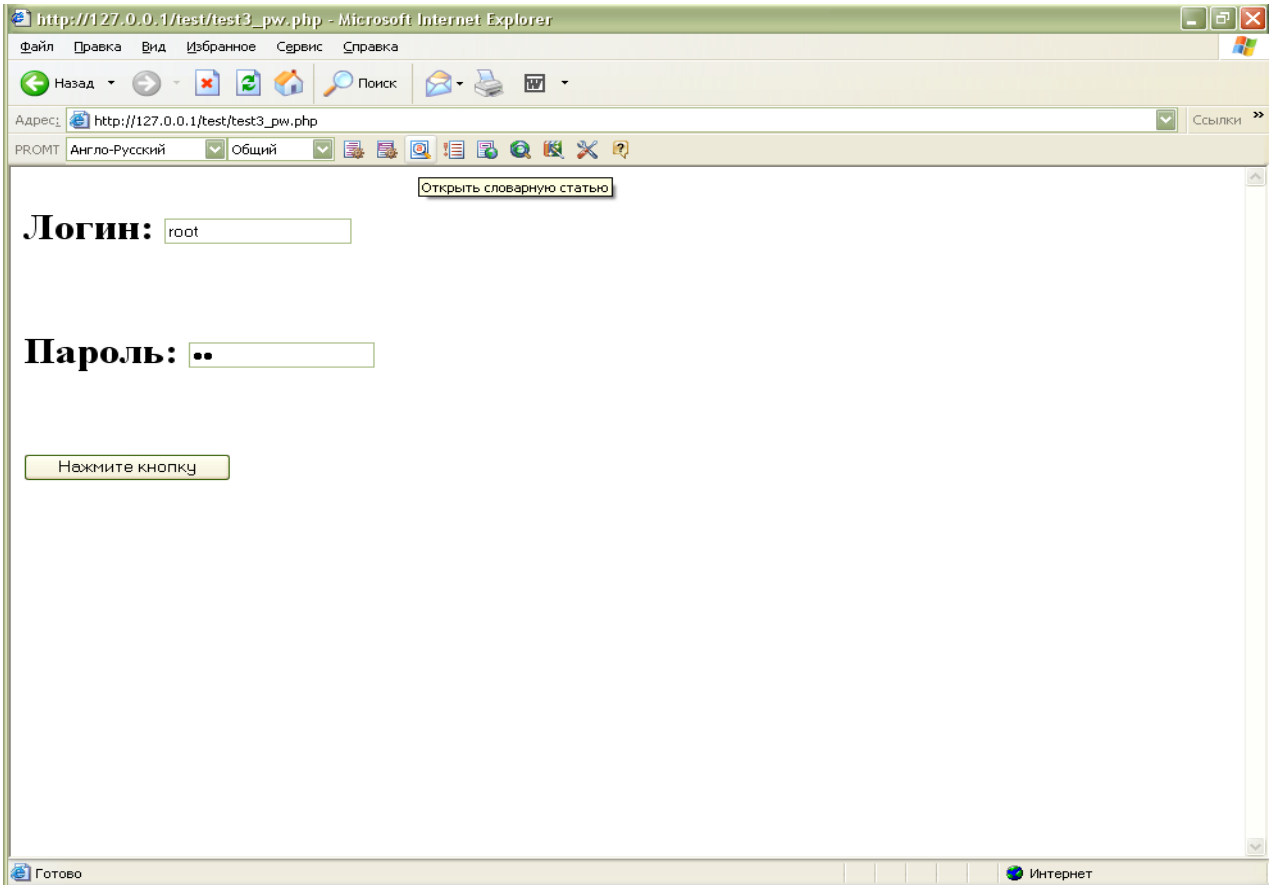


Рис. 1.26. Результат листинга 7

Если в адресной строке браузера задать:

http://localhost/hello.php?login=root&password=zz ,

получается передача параметров в командной строке.

Проанализировать переменную окружения *$QUERY_STRING*, которая доступна в *PHP*, можно под именем *$_SERVER[QUERY_STRING]* (см. листинг 8, рис. 1.27).

Листинг 8. Файл проверки параметров командной строки *test_pw.php*.

```
<html><body>
<?php
 Echo '<b><h1>Данные из командной строки:,<br><br>';
 echo $_SERVER[QUERY_STRING],'</h1></b><br>';
?>
</body></html>
```

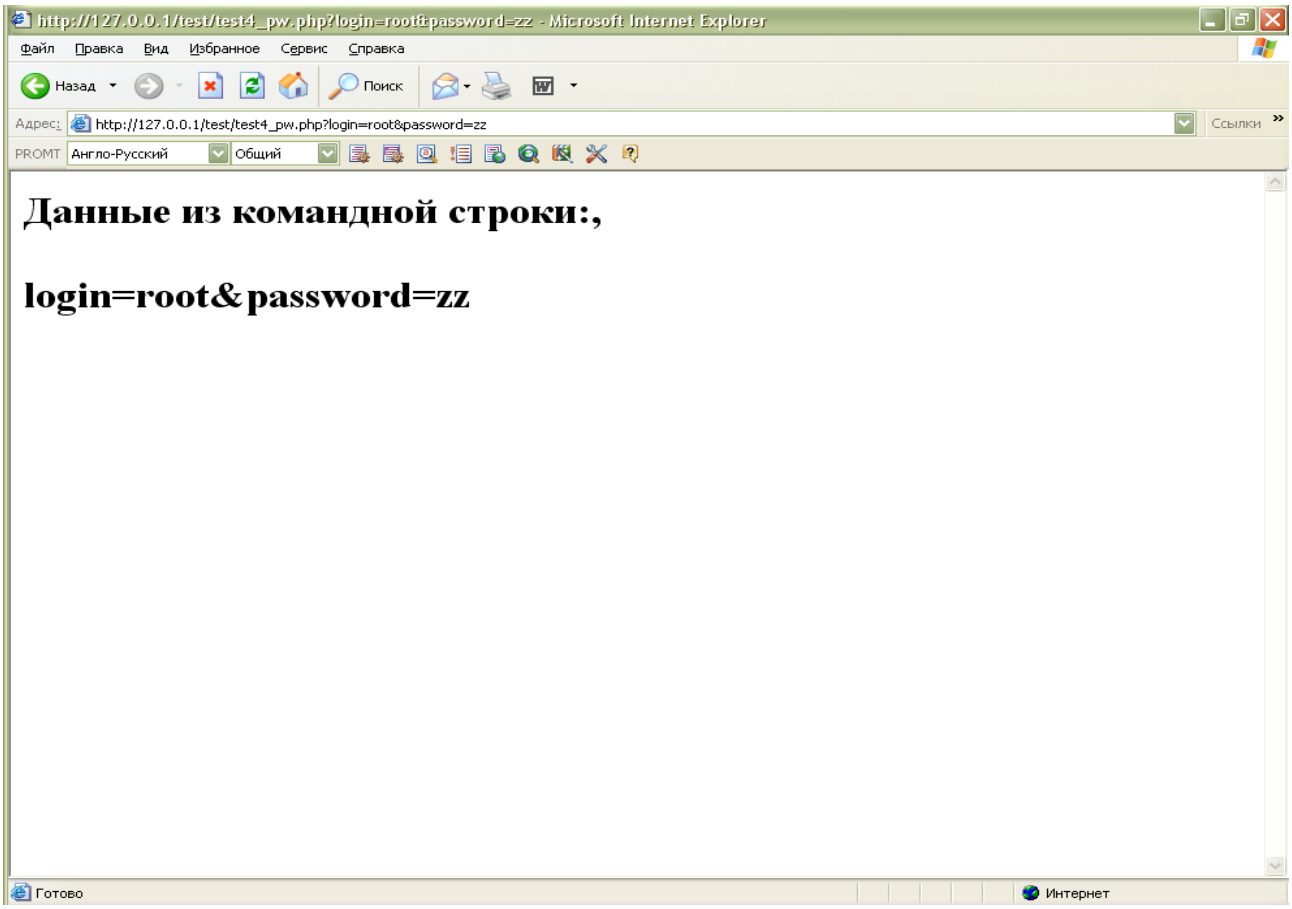


Рис. 1.27. Результат листинга 8

1.3.3.4. Трансляция полей формы

Все данные из полей формы *PHP* помещает в глобальный массив $_*REQUEST*. Значение поля *login* будет храниться в $_*REQUEST*[‘*login’*], а значение поля *password* будет храниться в $_*REQUEST*[‘*password*] (см. листинг 9, файл *test_pw.php*).

Листинг 9. Использование данных форм.

```
<html><body>
<?php
 if ($_REQUEST['login']=='root' && $_REQUEST['password']=='zz') {
        echo "Доступ открыт для пользователя - ";
        echo $_REQUEST['login'];
 } else {
        echo "Доступ закрыт";
 }
?>
</body></html>
```

1.3.3.5. Трансляция переменных окружения

Переменные окружения можно преобразовать в локальные переменные (листинг 10).

Листинг 10. Вывод *IP*-адреса и браузера пользователя.

```
<html><body>
<?PHP
  echo "Ваш IP-адрес:", $_SERVER['REMOTE_ADDR'],"<br>";
  echo "Ваш браузер:", $_SERVER['HTTP_USER_AGENT'];
?>
</body></html>
```

1.3.3.6. Трансляция cookies

Все переменные, переданные скрипту, создают массив (листинг 11).

Листинг 11. Демонстрация работы с $_*COOKIES*.

```
<?php
// В начале счетчик равен нулю
$count=0;
// Если в cookies что-то есть, берем счетчик оттуда.
If (isset ($_COOKIE['count'])){
  $count=($_COOKIE[count]);
  //echo "count==",$count;
  $count++;
  setcookie("count", $count,0x7FFFFFFF, '/');
}
// выводит счетчик
Echo "КУКА= ",$count;
?>
```

1.3.3.7. Обработка списков

В *PHP* предусмотрена возможность задавать имена полям формы в виде «массива с индексами» (листинг 12, рис. 1.28):

Листинг 12. Обработка списков.

```
<html>
   <head>
```

```
    <title>Пример</title>
  </head>
  <body>
    <?php
    echo '<FORM ACTION="test6_sel.php" METHOD=POST>
            Список городов:
            <SELECT NAME=gorod[]>
            <OPTION>Екатеринбург
            <OPTION>Москва
            <OPTION>Санкт-Петербург
            <OPTION>Киев
            </SELECT>
            <P>
            <INPUT TYPE=SUBMIT VALUE="Вывод">
        </FORM>';
 ?>
  </body>
</html>
```

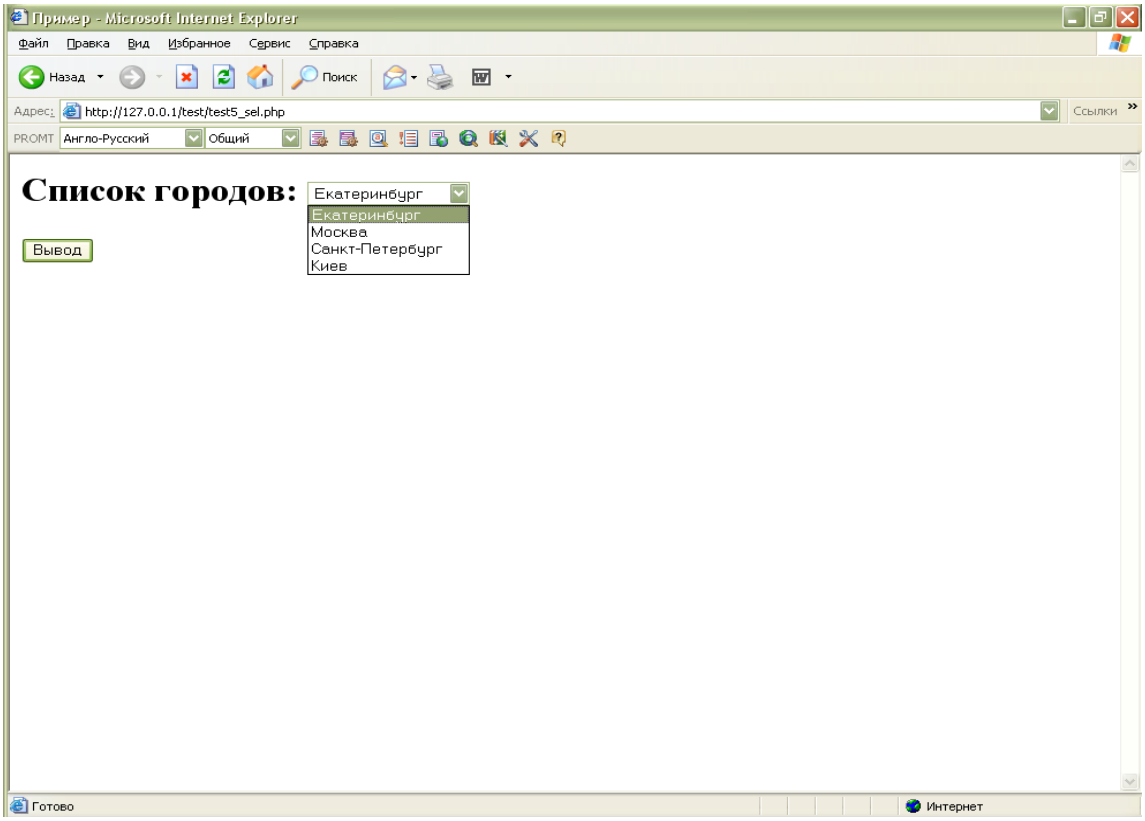


Рис. 1.28. Результат листинга 12

В результате получим массив в $_*REQUEST* массив массивов (двумерный), доступ к элементам которого можно получить так (листинг 13):

Листинг 13. *test6_sel.php.*

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?php
      $gor=$_REQUEST['gorod'][0];
      Echo $gor;  ?>
   </body>
</html
```

Пример ассоциативного массива представлен в файле *test7_sel.php* (листинг 14, рис. 1.29):

Листинг 14. Файл *test7_sel.php*.

```
<html>
  <head>
      <title>Пример</title>
  </head>
  <body>
    <?php
      echo '<FORM ACTION="test8_sel.php" METHOD=POST>
      Имя <input type=text name= Data[name]><br>
      Адрес <input type=text name= Data[address]><br>
      Город
      <input type=radio name= Data[city] value=Moscow>Москва<br>
      <input type=radio name= Data[city] value=Peter>Петербург<br>
      <input type=radio name= Data[city] value=kiev>Киев<br>
      <INPUT TYPE=SUBMIT VALUE="Вывод">
      </FORM>';
 ?>
  </body>
```

</html>

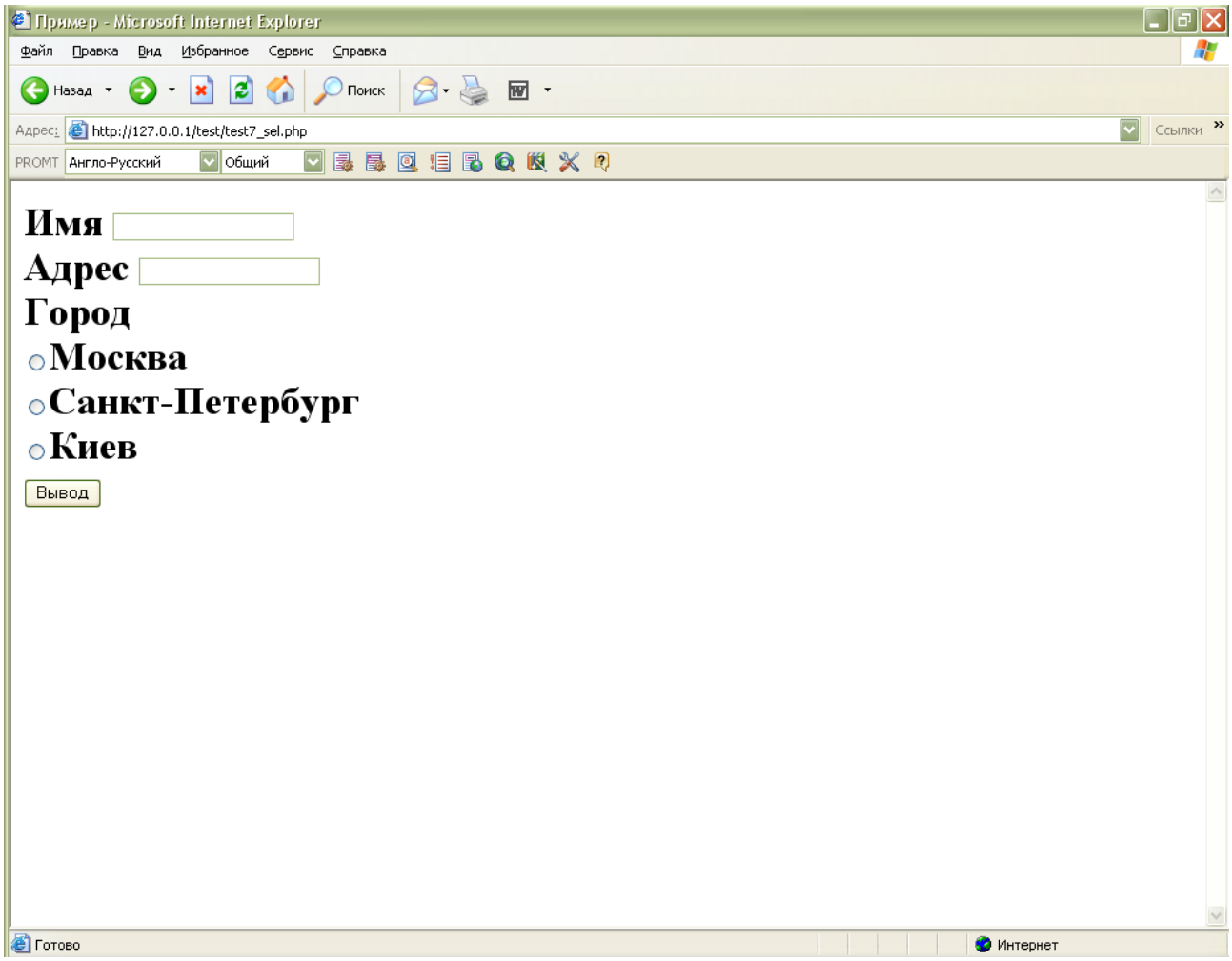


Рис. 1.29. Результат листинга 14

В сценарии к отдельным элементам формы можно обратиться при помощи указания ключа массива: например, *$REQUEST[‘Data’][‘city’]* (листинг 15):

Листинг 15. Файл *test8_sel.php*.

```
<html>
   <head>
      <title>Пример</title>
   </head>
   <body>
      <?php
                  $gor=$_REQUEST[Data][city];
                  Echo $gor;
                  echo '<br>',"Имя - ";
                  echo $_REQUEST[Data][name];
```

```
?>
  </body>
</html>
```

1.3.3.8. Передача параметров методами GET и POST

При использовании метода *GET* все параметры передаются единой строкой в переменной *QUERY_STRING*. Данные поступят *URL*-кодированными (см. подробнее в § 1.3.4, листинг 17).

При использовании метода *POST* параметры передаются сценарию через стандартный поток ввода (см. листинги 12, 14).

1.3.3.9. Особенности флажков checkbox

Можно воспользоваться одноименным скрытым полем (hidden) со значением, равным, например нулю, поместив его перед нужным флажком (см. листинг 16, рис. 1.30).

Листинг 16. Гарантированный прием значений от флажков в файле *test12_hid.php*.

```
<html>
 <head>
   <title>Пример</title>
</head>
<body>
 <?php
        foreach (@$_REQUEST[known] as $k=>$v) {
              If ($v) {echo "Вы знаете язык $k!"."<br>";}
              else {echo "Вы не знаете язык $k!"."<br>";}
        }
  echo '<form action=test12_hid.php method=POST>
  какие языки программирования вы знаете?<br>
  <input type=hidden name="known[PHP]" value="0">
  <input type=checkbox name="known[PHP]" value="1">PHP<br>
  <input type=hidden name="known[Perl]" value="0">
  <input type=checkbox name="known[Perl]" value="1">Perl<br>
  <input type=submit name="doGo" value="doGo">
  </form>';
```

```
?>
</body>
</html>
```

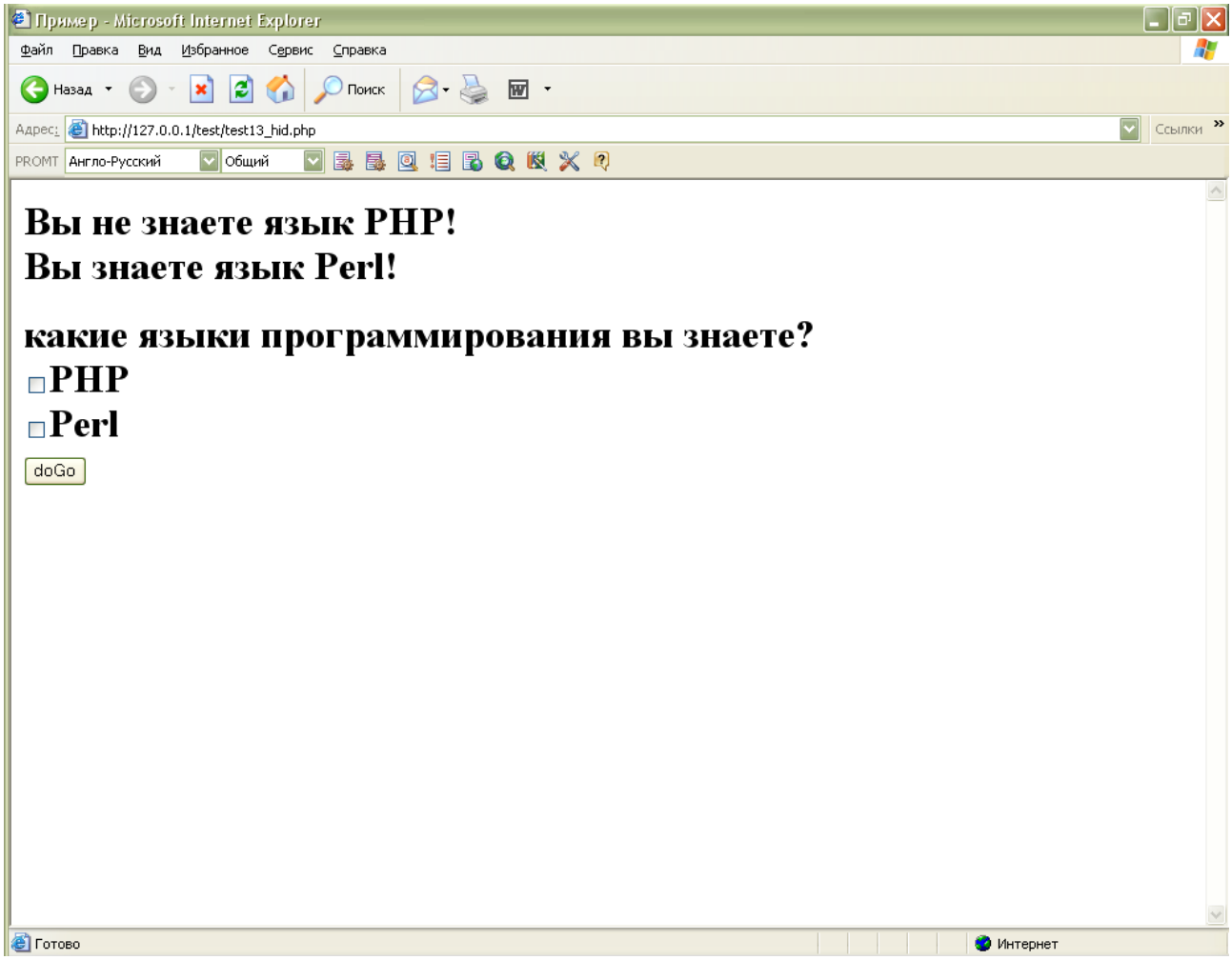


Рис. 1.30. Результат листинга 16

Если пользователь не выберет никакой флажок, браузер отправит сценарию пару *known*[язык] = 0 и в массиве $_*REQUEST*[‘*known’*] создастся соответствующий элемент. Если пользователь выберет флажок, то последует пара *know*[язык] = 1.

1.3.4. Пример формы

От описания базовых компонентов форм мы переходим к практическому примеру – построению формы для обработки данных, введенных пользователем. Допустим, вы хотите создать форму выбора периода времени (рис. 1.31).

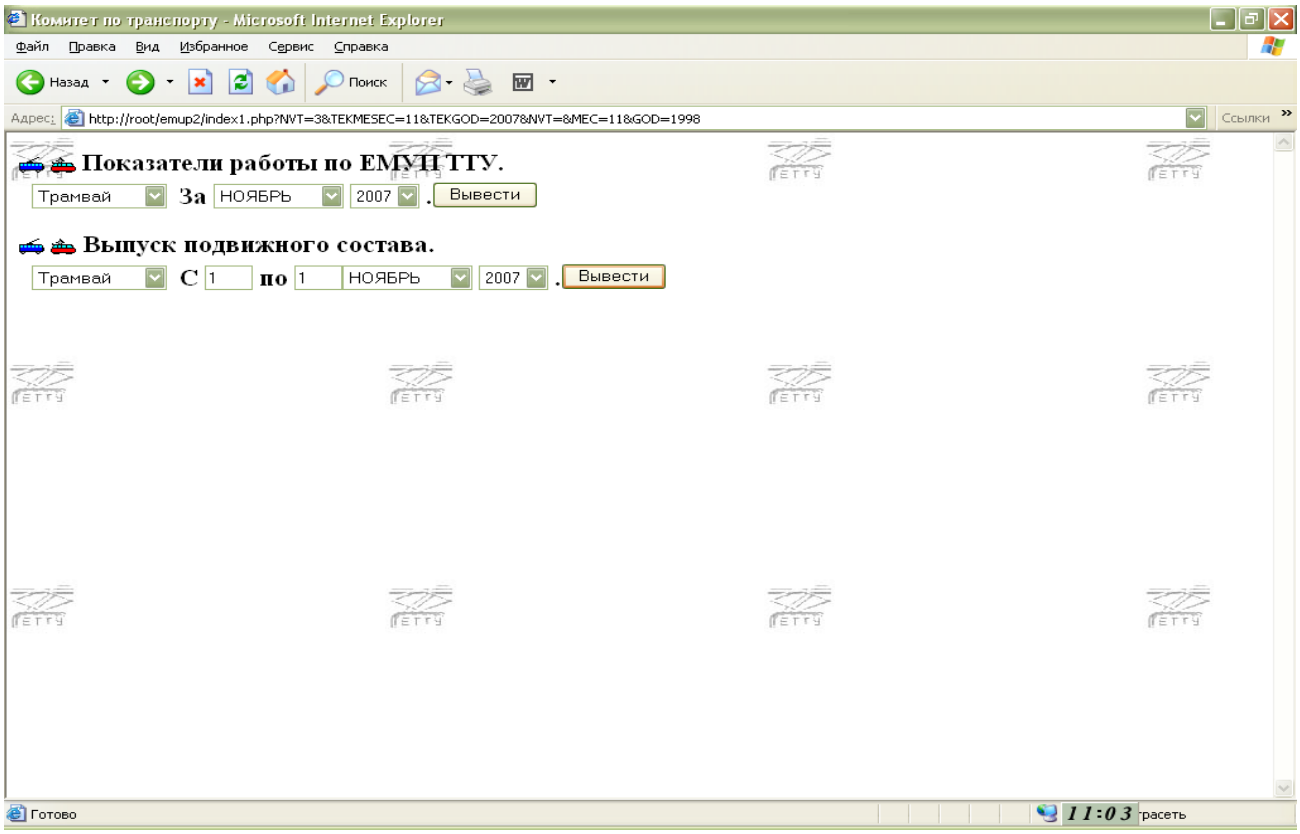


Рис. 1.31. Результат листинга 17

Программа «*index1.php*» приведена в листинге 17.

Листинг 17. Программа «*index1.php*»

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Комитет по транспорту</title>
</head>
<body bgcolor="#ffffff" BACKGROUND="images/fon.jpg">
<?Php
$DEN=date( 'd')+0; – объявление значений переменных
$MEC=date( 'm');
$GOD=date( 'Y');
Echo '<FORM METHOD="get" ACTION="Pokrab/pokrabttu_bas.php" target="_top">';
Echo '<IMG SRC="/image/troll.gif" BORDER=0 ALIGN=CENTER>';
Echo " ";
```

```
Echo "<B>Показатели работы по ЕМУП ТТУ.</B>";
Echo "<br>  ";
include ("../vvt2.php"); – список видов транспорта
Echo "<B> За ";
// месяцы года
Echo '<SELECT NAME="TEKMESEC">';
Echo '<OPTION ';if ($MEC==1) {Echo 'SELECTED';} Echo ' VALUE="01">
ЯНВАРЬ</OPTION>';
Echo '<OPTION ';if ($MEC==2) {Echo 'SELECTED';} Echo ' VALUE="02">
ФЕВРАЛЬ</OPTION>';
Echo '<OPTION ';if ($MEC==3) {Echo 'SELECTED';} Echo ' VALUE="03">
МАРТ</OPTION>';
Echo '<OPTION ';if ($MEC==4) {Echo 'SELECTED';} Echo ' VALUE="04">
АПРЕЛЬ</OPTION>';
Echo '<OPTION ';if ($MEC==5) {Echo 'SELECTED';} Echo ' VALUE="05">
МАЙ</OPTION>';
Echo '<OPTION ';if ($MEC==6) {Echo 'SELECTED';} Echo ' VALUE="06">
ИЮНЬ</OPTION>';
Echo '<OPTION ';if ($MEC==7) {Echo 'SELECTED';} Echo ' VALUE="07">
ИЮЛЬ</OPTION>';
Echo '<OPTION ';if ($MEC==8) {Echo 'SELECTED';} Echo ' VALUE="08">
АВГУСТ</OPTION>';
Echo '<OPTION ';if ($MEC==9) {Echo 'SELECTED';} Echo ' VALUE="09">
СЕНТЯБРЬ</OPTION>';
Echo '<OPTION ';if ($MEC==10) {Echo 'SELECTED';} Echo ' VALUE="10">
ОКТЯБРЬ</OPTION>';
Echo '<OPTION ';if ($MEC==11) {Echo 'SELECTED';} Echo ' VALUE="11">
НОЯБРЬ</OPTION>';
Echo '<OPTION ';if ($MEC==12) {Echo 'SELECTED';} Echo ' VALUE="12">
ДЕКАБРЬ</OPTION>';
Echo '</SELECT> ';
$ii = 1999; // годы
Echo '<SELECT NAME="TEKGOD">';
$i = $GOD;
while ($i >= $ii) {
Echo '<OPTION ';if ($GOD==$i) {Echo 'SELECTED ';}
```

```
Echo 'VALUE="'.$i.'">'.$i.'</OPTION>';
$i--;
}
Echo '</SELECT> ';
Echo '</B><INPUT TYPE="SUBMIT" VALUE="Вывести">';
Echo "</FORM>";
?>.
```

1.3.5. Задания для самостоятельной работы

1.3.5.1. План выполнения самостоятельной работы

Открыть пакет «*XAMPP Control Panel*». Ярлык на рабочем столе или в папке *C:/xampp/xampp-control.exe*. Запустить локальный *Web*-сервер *Apache* для *Windows* и *MySQL*.

Используя пакет «*NotePad++*», создайте первый сайт с расширением **.php*. Файл сохраните в папке, например *gruppa1* директории *C:/xampp/htdocs/*. Запустите его из адресной строки браузера, например:

по доменному адресу

«http://localhost/gruppa1/primer1.php »

или по IP-адресу

«http://127.0.0.1/gruppa1/primer1.php».

При отладке скрипта в браузере для обновления сайта используйте клавишу *F5* или кнопку «Обновить».

Создайте второй сайт из предложенного задания. Все данные из полей формы *PHP* помещайте в глобальный массив *$_REQUEST*.

1.3.5.2. Варианты заданий

Задание 1

Напишите *PHP*-программу создания формы «Ввод простого текста и его вывод».

На форме расположить:

- поле ввода (используйте тег <*input*>, параметр *type=text*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В поле ввода набивается фамилия, имя и отчество студента. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится:

- в первой строке фамилия студента «жирным» шрифтом;

- во второй строке – его имя шрифтом «курсив»;

- в третьей строке – фамилия и инициалы студента.

Примечание: функция длины строки *strlen*(*string* $*st*) *int*, где $*st* последовательность символов. Функция поиска подстроки *strpos*(*string* $*where*, *string* $*what*, *int* $*from*=0) *int*, где в строке $*where* ищется строка $*what*. В случае успеха эта функция возвращает позицию этой подстроки в строке. Функция *substr*(*string* $*str*, *int* $*start* [,*int* $*length*]) *string* возвращает часть строки $*str*, начиная с позиции $*start*, длиной $*length*.

Задание 2

Напишите *PHP*-программу создания формы «Ввод пароля и вывод проверки – правильно ли введен пароль».

На форме расположить:

- поле ввода (используйте тег <*input*>, параметр *type=password*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «ОК».

На кнопку «Вывести» наложите еще одну PHP-программу, с помощью которой в новой форме выводится сообщение о проверке правильности введенного пароля. На экране должно выйти сообщение «Пароль введен верно» или «Пароль введен неверно». Сообщение должно выводиться жирным шрифтом.

Задание 3

Напишите *PHP*-программу создания формы «Выбор параметра независимого переключателя и вывод соответствующего значения».

На форме расположить:

- независимый переключатель (или флажок) (используйте тег <*input*>, параметр *type=checkbox* поле ввода);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

Значения переключателя – «текущее время», «текущая дата», «текущий день», «текущий месяц», «текущий год». На кнопку «Вывести» наложите еще одну PHP-программу, с помощью которой в новой форме выведите выбранное значение. Этот параметр определяется по выбору переключателя.

Примечание: функция текущей даты *date*(“*d.m.y*”), функция времени *date*(“*h.i.s*”).

Задание 4

Напишите *PHP*-программу создания формы «Выбор параметра независимого переключателя и вывод соответствующего значения цвета».

На форме расположить:

- независимый переключатель (или радиокнопки) (используйте тег <*input*>, параметр *type=radio*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

Значения переключателя – «красный цвет», «желтый цвет», «синий цвет», «зеленый цвет». На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится текст названия цвета в соответствующей цветовой гамме. Название цвета выведите столбиком посередине формы.

Примечание: используйте тег *FONT,* который имеет следующий синтаксис: *<FONT SIZE=n>текст</FONT>* или *<FONT COLOR=цветовая спецификация>текст</FONT>*. Для переноса строки используется тег <*br*>.

Задание 5

Напишите *PHP*-программу создания формы «Вставка картинки и вывод даты или времени».

На форме расположить:

- два рисунка для отправки формы (*image*) (используйте тег <*input*>, параметр *type=image*).

На первый рисунок наложите программу вывода текущего времени, на второй рисунок программу вывода текущей даты. Перед значением текущей даты (например, 01.05.2009 г.) напишите слово «Сегодня 1 мая 2009 года». Перед текущим временем напишите «Текущее время» в верхнем регистре.

Примечание: функция *strtoupper*(*string* $*st*) *string* переводит строку в верхний регистр.

Задание 6

Напишите *PHP*-программу создания формы «Создание формы выбора даты и вывод этой даты».

На форме расположить:

- поле ввода значения числа дня (используйте тег <*input*>, параметр *type=text*);

- раскрывающийся список *select* значения названия месяца (используйте атрибуты <*name*>, <*option*> *и value*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится выбранная дата (например, 1 мая 2009 года).

Задание 7

Напишите *PHP*-программу создания формы «Созданные формы выбора натурального числа и вывод квадрата или куба этого числа».

На форме расположить:

- раскрывающийся список *select* значения названия натурального числа 1 до 10 (используйте атрибуты *<name>, <option> и value*);

- независимый переключатель (или радиокнопка) (используйте тег *<input>*, параметр *type=radio*);

- кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

На радиокнопку наложить два параметра: «квадрат», «куб». На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится квадрат или куб выбранного натурального числа соответственно выбранному параметру радиокнопки.

Примечание: для возведения в квадрат используйте функцию *pow(float $base, float $exp) float* возвращает $*base* в степень $*expl*. Для возведения в куб – используйте цикл *for…end*.

Задание 8

Напишите *PHP*-программу создания формы «Ввод интервала простых чисел и вывод случайного числа из этого интервала».

На форме расположить:

- два поля ввода (используйте тег *<input>*, параметр *type=text*);

- кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

В первом поле ввода набивается начальное значение интервала, во втором поле ввода – конечное значение интервала. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится случайное число.

Примечание: функция возвращения случайных чисел *mt_rand(int $min=0, int $max=RAND_MAX) int.*

Задание 9

Напишите PHP-программу создания формы «Создание массива фамилий и демонстрация работы со списками».

На форме расположить:

- кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится список фамилий. Список пронумеруйте.

Примечание: для вывода используйте цикл *for*. Количество элементов в массиве определяется функцией *count()*. Элементы массива пропишите в *PHP*-программе.

Задание 10

Напишите *PHP*-программу создания формы «Ввод интервала простых чисел и вывод списка заданных значений».

На форме расположить:

- два поля ввода (используйте тег <*input*>, параметр *type=text*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В первом поле ввода вводится начальное значение интервала, во втором поле ввода – конечное значение интервала. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится столбиком заданный интервал чисел, начиная со второго значения и заканчивая предпоследним значением.

Примечание: для вывода чисел напишите функцию вывода.

Задание 11

Напишите *PHP*-программу создания формы «Ввод значений любых чисел и вывод значения».

На форме расположить:

- два поля ввода (используйте тег <*input*>, параметр *type=text*);

- кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В первом и во втором поле ввода вводится дробное значение. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится результат перемножения этих чисел. А также выведите: ближайшее целое число, наименьшее целое число, максимальное целое число и абсолютное значение.

Примечание: функция *abs(mixed $numeric) mixed* возвращает модуль числа. Функция *round(double $val) double* округляет $*val* до ближайшего целого числа. Функция *ceil(float $number) int* возвращает наименьшее целое число, которое не меньше, чем *number*. Функция *float(float $number) int* возвращает максимальное целое число, не превосходящее *number*.

1.4. Разработка PHP-приложений с использованием баз данных

PHP (*Hypertext Preprocessor*) это широко распространённый открытый ресурс – язык скриптинга (сценариев) общего назначения, который создан специально для *Web* и который можно внедрять в *HTML*. *PHP* – это система разработки скриптов, включающая в себя *CGI*-интерфейс, интерпретатор языка

и набор функций для доступа к базам данных и различным объектам *WWW*. Он способен выполнять доступ к файлам, исполнять команды и открывать сетевые соединения на сервере. *PHP* используется на всех крупных операционных системах (ОС), включая *Linux*. *PHP* имеет поддержку для большинства существующих *web*-серверов, например для *Apache*-сервера. *PHP* поддерживает большое количество баз данных (БД). *PHP* позволяет написать фрагмент следующего вида (см. листинг 18):

Листинг 18

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?php
    echo «Hi, I'm a PHP script!»;
    ?>
  </body>
</html>
```

Код *PHP* заключён в специальные начальный и конечный тэги, что позволяет входить и выходить из «режима *PHP*». *PHP* отличается от других подобных языков тем, что код выполняется на сервере. Система *PHP* устанавливается на *Web*-сервер в качестве обработчика определенного *mime* – типа, аналогично системе *SSI* (*Server Side Include*), то есть *PHP*-скрипт представляет собой обычный *html*-документ с редкими вставками тэгов *PHP*-команд.

Самая значимая возможность в *PHP* – уровень интеграции с базами данных**.**

1.4.1. Синтаксис и грамматика

1.4.1.1. Переход из HTML

Есть три способа выхода из HTML и перехода в «режим PHP кода»: Способы перехода из HTML:

1. <? *echo*(«простейший способ, инструкция обработки *SGML**n*»); ?>

2. <?*php echo*(«при работе с *XML* документами делайте так*n*»); ?>

3. <*script language*=«*php*»>*echo* («некоторые редакторы (подобные *FrontPage*) не любят обрабатывающие инструкции»);</*script*>;

4. <% *echo*(«От *PHP* 3.0.4 можно факультативно применять *ASP*-тэги»); %>

1.4.1.2. Разделение инструкций

Инструкции (утверждения) разделяются точкой с запятой. Закрывающий тэг (?>) тоже подразумевает конец утверждения, поэтому следующие записи эквивалентны:

<php

echo «Это тест»;

?>

<php echo «Это тест»?>

1.4.2. Общие сведения о формах

При вводе данных в форму используются различные управляющие элементы. В одних элементах пользователь вводит информацию с клавиатуры, в других он выбирает нужный вариант, щелкая кнопкой мыши. В формах могут присутствовать скрытые поля, которые поддерживаются самой формой; содержимое скрытых полей не должно изменяться пользователем.

Одна страница может содержать несколько форм, поэтому необходимы средства, которые позволяли бы отличить одну форму от другой. Более того, как-то требуется сообщить форме, куда следует перейти, когда пользователь выполняет действие с формой (как правило, нажимает кнопку отправки данных). Обе задачи решаются заключением форм в следующие теги *HTML*:

<form action = действие method = «метод» – элементы формы – </form>.

Как видно из приведенного фрагмента, в тегах форм указываются два важных элемента: действие и метод. Действие указывает, какой сценарий должен обрабатывать форму, а метод определяет способ передачи данных этому сценарию. Существует два метода:

Метод *get* передает все данные формы в конце *URL*. Из-за различных ограничений, связанных со спецификой языков и длиной данных, этот метод применяется редко.

Метод *post* передает все данные формы в теле запроса. Этот метод используется чаще, чем *get*.

Элементов, ориентированных на ввод с клавиатуры, всего два – текстовое поле (*text box*) и текстовая область (*text area*).

1.4.2.1. Текстовое поле

В текстовых полях обычно вводится короткая текстовая информация. Синтаксис определения текстового поля:

<input type = «text» name = «имя_переменной» size = «N» maxlength = «N» value = « »>.

Определение текстового поля включает пять атрибутов:

type – тип элемента (для текстовых полей – *text*);

name – имя переменной, в которой сохраняются введенные данные;

size – общий размер текстового поля в браузере;

maxlength – максимальное количество символов, вводимых в текстовом поле;

value – значение, отображаемое в текстовом поле по умолчанию.

Текстовое поле изображено на рис. 1.32.

Рис. 1.32. Текстовое поле

Особой разновидностью текстовых полей является поле для ввода паролей. Оно работает точно так же, как обычное текстовое поле, однако вводимые символы заменяются звездочками. Чтобы создать в форме поле для ввода паролей, достаточно указать *type=«password»* вместо *type=«text»*.

В других элементах форм пользователь выбирает один из заранее определенных вариантов при помощи мыши. Это флажки, переключатели и раскрывающиеся списки.

1.4.2.2. Флажок

Флажки (*checkboxes*) используются в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора – по аналогии с тем, как ставятся «галочки» в анкетах. Синтаксис определения флажка:

<input type = «checkbox» name = «имя_переменной» value = «начальное_значение»>.

Определение флажка включает три атрибута:

type – тип элемента (для флажков – *checkbox*);

name – имя переменной, в которой сохраняются введенные данные (в данном случае – состояние элемента);

value – значение, присваиваемое переменной по умолчанию. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута *value* не используется.

Флажок изображен на рис. 1.33.

Рис. 1.33. Флажок

1.4.2.3. Переключатель

Переключатель (*radio button*) представляет собой разновидность флажка – в любой момент времени в группе может быть установлен лишь один переключатель. Синтаксис определения переключателя:

<*input type=«radio» name=«имя_переменной» value=«начальное_значение»*>.

Определение переключателя поля включает три атрибута:

type – тип элемента (для переключателей – *radio*);

name – имя переменной, в которой сохраняются введенные данные (в данном случае – состояние элемента);

value – значение, присваиваемое переменной по умолчанию. Если переключатель установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута *value* не используется.

Переключатель изображен на рис. 1.34.

Рис. 1.34. Переключатель

1.4.2.4. Раскрывающийся список

Раскрывающийся список особенно удобен в ситуации, когда у вас имеется длинный перечень допустимых вариантов. Синтаксис определения раскрывающегося списка:

<*select name=«имя_переменной»*>
<*option value=«имя_переменной*1»>
<*option value=«имя_переменной*2»>
<*option value=«имя_переменной*3»>

<option value=«имя_переменнойN»>

</select>

Определение переключателя поля включает три атрибута:

name – имя переменной, в которой сохраняются введенные данные (в данном случае – строка, выбранная в списке);

value – значение, отображаемое в списке по умолчанию.

Раскрывающийся список изображен на рис. 1.35.

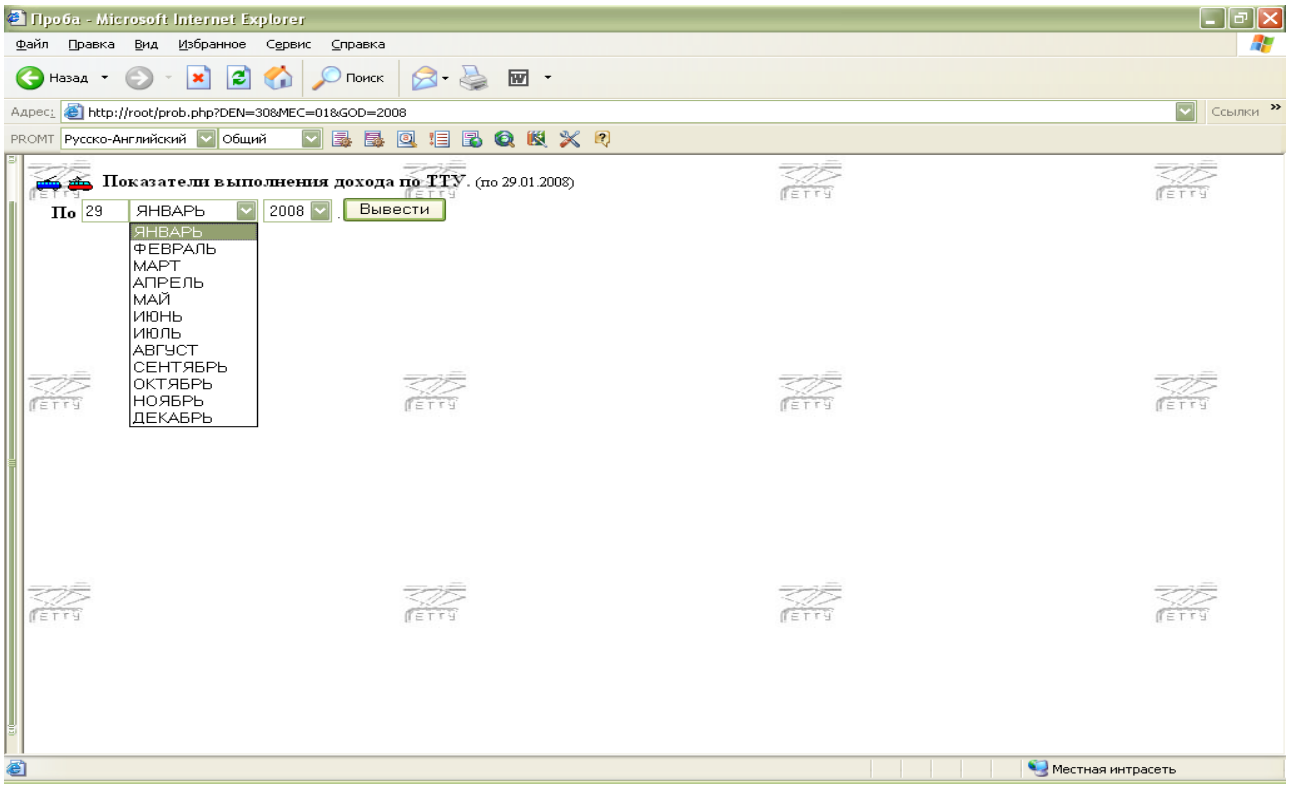


Рис. 1.35. Раскрывающийся список

1.4.2.5. Скрытые поля

Скрытые поля не отображаются в браузере и обычно используются для передачи данных между сценариями. Синтаксис определения скрытого поля практически идентичен синтаксису текстовых полей, отличается только атрибут типа. Поскольку скрытые поля не отображаются в браузере, привести пример на страницах книги невозможно. Синтаксис определения скрытого поля:

<input type=«hidden» name=«имя_переменной» value=«начальное_значение»>.

Определение скрытого поля включает три атрибута:

type – тип элемента (для скрытых полей – *hidden*);

name – имя переменной, в которой сохраняются скрытые данные;

value – значение, по умолчанию сохраняемое в скрытом поле.

1.4.2.6. Кнопка отправки данных

Кнопка отправки данных инициирует действие, заданное атрибутом *action* тега *<form>* (рис. 1.36). Синтаксис определения:

<input type=«submit» value=«текст_на_кнопке»>.

Определение кнопки включает два атрибута:

type – тип элемента (для кнопки отправки данных – *submit*);

value – текст, по умолчанию отображаемый на кнопке.

Рис. 1.36. Кнопка отправки данных

1.4.2.7. Кнопка сброса

Кнопка сброса отменяет все изменения, внесенные в элементы формы. Синтаксис определения:

<input type=«reset» value=«текст _на_кнопке»>.

Определение кнопки включает два атрибута:

type – тип элемента (для кнопки сброса – *reset*);

value – текст, по умолчанию отображаемый на кнопке.

1.4.3. Что такое SQL?

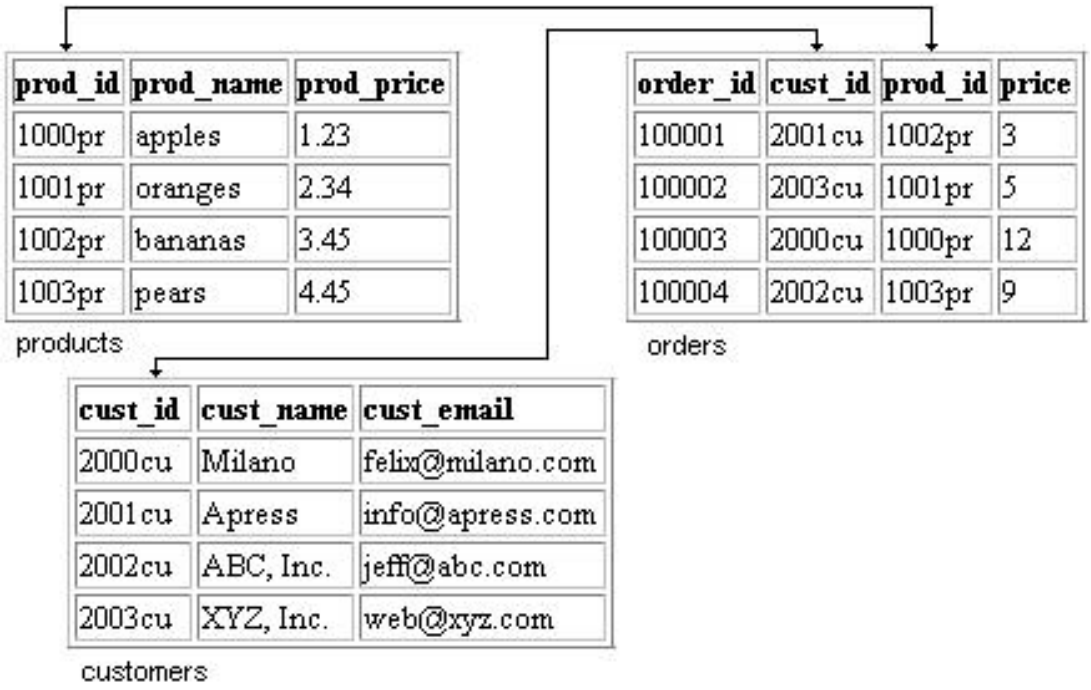
SQL (*Structured Query Language*) – язык запросов для динамического представления содержимого базы данных.

SQL обычно описывается как стандартный язык, используемый для взаимодействия с реляционными базами данных. Это интерфейсное средство для выполнения различных операций с базами данных, предоставляющее в распоряжение пользователя стандартный набор команд. Возможности SQL не ограничиваются выборкой данных из базы. Для получения или сохранения любой информации вам необходимо установить соединение с БД, отправить верный запрос, получить результат и закрыть соединение.

В *SQL* поддерживаются разнообразные возможности для взаимодействия с базой данных, в том числе:

- определение структуры данных – определение конструкций, используемых при хранении данных;
- выборка данных – загрузка данных из базы и их представление в формате, удобном для вывода;
- обработка данных – вставка, обновление и удаление информации;
- контроль доступа – возможность разрешения/запрета выборки, вставки, обновления и удаления данных на уровне отдельных пользователей;
- контроль целостности данных – сохранение структуры данных при возникновении таких проблем, как параллельные обновления или системные сбои.

В реляционных СУБД данные организуются в виде набора взаимосвязанных таблиц. Связи между таблицами реализуются в виде ссылок на данные других таблиц. Таблицу можно представить себе как двухмерный массив, в котором расположение каждого элемента характеризуется определенными значениями строки и столбца. Пример реляционной базы данных изображен на рис. 1.37.



products

prod_id	prod_name	prod_price
1000pr	apples	1.23
1001pr	oranges	2.34
1002pr	bananas	3.45
1003pr	pears	4.45

orders

order_id	cust_id	prod_id	price
100001	2001cu	1002pr	3
100002	2003cu	1001pr	5
100003	2000cu	1000pr	12
100004	2002cu	1003pr	9

customers

cust_id	cust_name	cust_email
2000cu	Milano	felix@milano.com
2001cu	Apress	info@apress.com
2002cu	ABC, Inc.	jeff@abc.com
2003cu	XYZ, Inc.	web@xyz.com

Рис. 1.37. Пример реляционной базы данных

Как видно из рис. 1.37, каждая таблица состоит из строк (записей) и столбцов (полей). Каждому полю присваивается уникальное (в рамках данной таблицы) имя. Обратите внимание на связь между таблицами *customer* и *orders*, обозначенную стрелкой. В информацию о заказе включается короткий идентификатор клиента, что позволяет избежать избыточного хранения имени и прочих реквизитов клиента. В изображенной базе данных существует еще одна связь – между таблицами *orders* и *products*. Эта связь устанавливается по полю *prod_id*, в котором хранится идентификатор товара, заказанного данным клиентом (определяемого полем *custjd*). Наличие этих связей позволяет легко ссылаться на полные данные клиента и товара по простым идентификаторам. Правильно организованная база данных превращается в мощное средство организации и эффективного хранения данных с минимальной избыточностью.

1.4.4. База данных MySQL

MySQL – это система управления базами данных.

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми – от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных,

хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО *MySQL*. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному – как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

MySQL – это система управления реляционными базами данных.

В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. *SQL* как часть системы *MySQL* можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

Программное обеспечение *MySQL* – это программное обеспечение с открытым кодом.

1.4.4.1. Синтаксис оператора SELECT

Команда *SELECT* предназначена для извлечения строк данных из одной или нескольких таблиц. Оператор *SELECT* имеет следующую структуру:

SELECT [STRAIGHT_JOIN]

[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]

[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]

[DISTINCT | DISTINCTROW | ALL]

select_expression,...

[INTO {OUTFILE | DUMPFILE} «*file_name*» *export_options]*

[FROM table_references

[WHERE where_definition]

[GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]

[HAVING where_definition]

[ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]

[LIMIT [offset,] rows | rows OFFSET offset]

[PROCEDURE procedure_name(argument_list)]

[FOR UPDATE | LOCK IN SHARE MODE]].

SELECT применяется для извлечения строк, выбранных из одной или нескольких таблиц. Выражение *select_expression* задает столбцы, в которых необходимо проводить выборку.

Здесь *COLUMN* – имя выбираемого столбца. Можно указать несколько столбцов через запятую. Если необходимо выбрать все столбцы, можно просто указать знак звёздочки *. Ключевое слово *FROM* указывает таблицу *TABLE*, из которой извлекаются записи. Ключевое слово *WHERE*, так же как и в операторе *DELETE*, определяет условия отбора строк. Ключевое слово *ORDER BY* сортирует строки запросов по столбцу *COL_NAME* в прямом (*ASC*) или обратном порядке (*DESC)*. Ключевое слово *LIMIT* сообщает *MySQL* о выводе только *ROWS* запросов, начиная с позиции *OFFSET*.

Кроме того, оператор *SELECT* можно использовать для извлечения строк, вычисленных без ссылки на какую-либо таблицу. Например:

mysql> *SELECT* 1 + 1;

Результат -> 2

1.4.4.2. Синтаксис оператора INSERT

Оператор *INSERT* вставляет новые строки в существующую таблицу.

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl_name [(col_name,...)]

VALUES (expression,...),(...),...

[ON DUPLICATE KEY UPDATE col_name=expression, ...]

или

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl_name [(col_name,...)]

SELECT ...

или

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl_name

SET col_name=(expression | DEFAULT), ...

[ON DUPLICATE KEY UPDATE col_name=expression, ...].

Форма данной команды *INSERT ... VALUES* вставляет строки в соответствии с точно указанными в команде значениями. Форма *INSERT ... SELECT* вставляет строки, выбранные из другой таблицы или таблиц.

1.4.4.3. Синтаксис оператора UPDATE

Оператор *UPDATE* обновляет столбцы в соответствии с их новыми значениями в строках существующей таблицы.

UPDATE [LOW_PRIORITY] [IGNORE] tbl_name

SET col_name1=expr1 [, col_name2=expr2 ...]

[WHERE where_definition]

[ORDER BY ...]

[LIMIT rows]

или

UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, tbl_name ...]

SET col_name1=expr1 [, col_name2=expr2 ...]

[WHERE where_definition].

В выражении данного оператора *SET* указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении *WHERE*, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Если задано выражение *ORDER BY*, то строки будут обновляться в указанном в нем порядке.

1.4.4.4. Синтаксис оператора DELETE

Оператор *DELETE* удаляет из таблицы *TABLE_NAME* строки, удовлетворяющие заданным в *WHERE_DEFINITION* условиям, и возвращает число удаленных записей.

DELETE [LOW_PRIORITY] [QUICK] FROM table_name

[WHERE where_definition]

[ORDER BY ...]

[LIMIT rows]

или

DELETE [LOW_PRIORITY] [QUICK] table_name[.] [, table_name[.*] ...]*

FROM table-references

[WHERE where_definition]

или

DELETE [LOW_PRIORITY] [QUICK]

FROM table_name[.] [, table_name[.*] ...]*

USING table-references

[WHERE where_definition]

Если оператор *DELETE* запускается без определения *WHERE*, то удаляются все строки.

1.4.5. Стандартные функции PHP для работы с MYSQL

В этом разделе описаны стандартные функции *PHP*, при помощи которых можно организовать взаимодействие сценариев *PHP* с сервером *MySQL* (см. листинг 19). Общая последовательность действий при взаимодействии с сервером *MySQL* выглядит так:

1. Установить соединение с сервером *MySQL*. Если попытка завершается неудачей, необходимо вывести соответствующее сообщение и завершить процесс.
2. Выбрать базу данных сервера *MySQL*. Если попытка выбора завершается неудачей, вывести соответствующее сообщение и завершить процесс. Допускается одновременное открытие нескольких баз данных для обработки запросов.
3. Обработать запросы к выбранной базе (или базам).
4. После завершения обработки запросов закрыть соединение с сервером баз данных.

Листинг 19. Пример сценария *PHP* с базой данных.

```
<?Php
$sock=mysql_connect(«localhost», «nobody», «sasnobody»);
mysql_select_db(«disr»,$sock);
$SQL=«
SELECT VT AS TRP, VIDTRAN AS ID FROM disr.NACHDAN where vt>1 GROUP BY VT ORDER BY VT DESC
»;
$R=mysql_query($SQL,$sock);
$T=mysql_fetch_array($R);
Echo «<SELECT NAME=«NVT»>»;
while(is_array($T))
{
Echo «<OPTION VALUE=».$T[TRP]. «>».$T[ID];
$T=mysql_fetch_array($R);
}
Echo «</SELECT>»;
$sock=mysql_close;
?>.
```

1.4.5.1. Mysql_connect()

Функция *mysql_connect*() устанавливает связь с сервером *MySQL*. После успешного подключения к *MySQL* можно переходить к выбору баз данных, обслуживаемых этим сервером. Синтаксис функции:

mysql_connect ([string хост [:порт] [:/путь//к/сокету] [, string имя пользователя] [, string пароль])

В параметре хост передается имя хостового (гостевого) компьютера, указанное в таблицах привилегий сервера *MySQL*. Конечно, оно же используется для перенаправления запросов на *web*-сервер, на котором работает *MySQL*, поскольку к серверу *MySQL* можно подключаться в удаленном режиме. Наряду с именем хоста могут указываться необязательные параметры – номер порта, а также путь к сокету (для локального хоста). Параметры имя_пользователя и пароль должны соответствовать имени пользователя и паролю, заданным в таблицах привилегий *MySQL*. Пример открытия соединения с *MySQL*:

@mysql_connect(«localhost»,«nobody»,«sasnobody») or die(«Could not connect to MySQL server! »).

В данном примере *localhost* – имя компьютера, *nobody* – имя пользователя, а *sasnobody* – пароль. Знак @ перед вызовом функции *mysql_connect*() подавляет все сообщения об ошибках, выдаваемые при неудачной попытке подключения, – они заменяются сообщением, указанным при вызове *die*().

1.4.5.2. Mysql_select_db()

После успешного соединения с *MySQL* необходимо выбрать базу данных, находящуюся на сервере. Для этого используется функция *mysql_select_db()*. Синтаксис функции:

mysql_select_db (string имя_базы_данных [, int идентификатор_соединения]).

Параметр *имя_базы_данных* определяет выбираемую базу данных и идентификатор, который возвращается функцией *mysql_select_db()*. Пример выбора базы данных функцией:

<? Php

$sock=mysql_connect(«localhost»,«nobody»,«sasnobody») or die(«Could not connect to MySQL server!»);

mysql_select_db(«godar»,$sock) or die(«Could not select company database!»);

?>

1.4.5.3. Mysql_close()

После завершения работы с сервером *MySQL* соединение необходимо закрыть. Функция *mysql_close()* закрывает соединение, определяемое

необязательным параметром. Если параметр не задан, функция закрывает последнее открытое соединение. Синтаксис функции:

mysql_close ([int идентификатор_соединения]).

Пример использования *mysql_close*():

<?

$sock=mysql_connect(«localhost»,«nobody»,«sasnobody») or die(«Could not connect to MySQL server!»);

mysql_select_db(«godar»,$sock) or die(«Could not select company database!»);

print «You're connected to a MySQL database! »;

mysql_close();

?>

1.4.5.4. Mysql_query()

Функция *mysql_query()* обеспечивает интерфейс для обращения с запросами к базам данных. Синтаксис функции:

mysql_query (string запрос [, int идентификатор_соединения]).

Параметр *запрос* содержит текст запроса на языке *SQL*. Если обработка запроса завершилась неудачей, функция возвращает *FALSE*. Пример использования функции:

$SQL=«

select TN,FIO,CLASS from nariyd_new.FIO

*Where TP=*301 *And TIP=*1»*;*

$Result=mysql_query($SQL,$sock);

1.4.5.5. Mmysql_num_rows()

Функция *mysql_num_rows()* определяет количество записей, возвращаемых командой *SELECT*. Синтаксис функции*:*

int mysql_num_rows(int результат).

Пример использования функции *mysql_num_rows()*:

<?

// Подключиться к серверу и выбрать базу данных

@mysql_connect(«localhost»,«web»,«4tf9zzzf») or die(«Could not connect to MySQL server! »);

@mysql_select_db(«company») or die(«Could not select company database! »);

// Выбрать все товары, названия которых начинаются с «*p*»

$query = «SELECT prod_name FROM products WHERE prod_name LIKE

«*p%*» »*;*

// Выполнить запрос

$result = mysql_query($query);

print «*Total rows selected:*».*mysql_num_rows($result);*

mysql_close();

?>

Поскольку таблица содержит лишь один товар, название которого начинается с буквы *p* (*pears*), возвращается только одна запись. Результат:

Total rows selected: 1

1.4.5.6. Mysql_result()

Функция *mysql_result()* используется в сочетании с *mysql_query()* (при выполнении запроса с командой *SELECT*) для получения набора данных. Синтаксис функции:

*mysql_result (int идентификатор_результата, int запись [, mixed поле])***.**

В параметре *идентификатор_результата* передается значение, возвращенное функцией *mysql_query()*. Параметр *запись* ссылается на определенную запись набора данных, определяемого параметром *идентификатор_результата*.

Наконец, в необязательном параметре *поле* могут передаваться:

- смещение поля в таблице;
- имя поля;
- имя поля в формате *имя_поля_имя_таблицы*.

В листинге 20 используется база данных *nariyd_new* и таблица *FIO* (фамилии).

Листинг 20. Выборка и форматирование данных в базе данных *MySQL***.**

<?

$sock=mysql_connect(«*localhost*»,«*nobody*»,«*sasnobody*»*) or die(*«*Could not connect to MySQL server!*»*);*

mysql_select_db(«*godar*»*,$sock) or die(*«*Could not select company database!*»*);*

$SQL=«

select TN,FIO,CLASS from nariyd_new.FIO

Where TP=301 *And TIP*=1»

$Result=mysql_query($SQL,$sock);

$kolzap=mysql_numrows($Result);

```
Echo «Список фамилий»;
for ($j=0;$j<=$kolzap-1;$j++) {
  Echo $j+1, «. »;
  Echo mysql_result($RF,$j, «FIO»)). «; »;
  Echo «<br>»;
}
mysql_close();
?>
```

В результате выполнения этого примера будет получен следующий результат:

Список фамилий

Иванов;

Петров;

Синицына;

1.4.5.7. Mysql_fetch_row()

Обычно гораздо удобнее сразу присвоить значения всех полей записи элементам индексируемого массива (начиная с индекса 0), нежели многократно вызывать *mysql_result()* для получения отдельных полей. Задача решается функцией *mysql_fetch_row()*, имеющей следующий синтаксис:

array mysql_fetch_row (int результат).

Использование функции *list*() в сочетании с *mysql_fetch_row()* позволяет сэкономить несколько команд, необходимых при использовании *mysql_result().* В листинге 21 приведен код листинга 20, переписанный с использованием *list*() и *mysql_fetch_row().*

Листинг 21. Выборка данных функцией *mysql_fetch_row().*

```
<? php
$sock=mysql_connect(«localhost»,«nobody»,«sasnobody») or die(«Could not connect to MySQL server!»);
mysql_select_db(«godar»,$sock) or die(«Could not select company database!»);
$SQL=«
  select TN,FIO,CLASS from nariyd_new.FIO
  Where TP=301 And TIP=1»;
$Result=mysql_query($SQL,$sock);
$i=0;
while ($row = mysql_fetch_row ($Result));
```

```
  Echo $i++;
  Print $row[«FIO»], «;», «<br>»;
  $row++;
endwhile;
mysql_close();
?>
```

Листинг 21 выдает тот же результат, что и листинг 20, но использует при этом меньшее количество команд.

1.4.5.8. Mysql_fetch_array()

Функция *mysql_fetch_array()* аналогична *mysql_fetch_row()*, однако по умолчанию значения полей записи сохраняются в ассоциативном массиве. Впрочем, вы можете выбрать тип индексации (ассоциативная, числовая или комбинированная). Синтаксис функции:

array mysql_fetch_array (int идентификатор результата [, тип_индексации]).

В параметре *идентификатор_результата* передается значение, возвращенное функцией *mysql_query().* Необязательный параметр *тип_индексации* принимает одно из следующих значений:

MYSQL_ASSOC – функция *mysql_fetch_array*() возвращает ассоциативный массив. Если параметр не указан, это значение используется по умолчанию;

MYSQL_NUM – функция *mysql_fetch_array*() возвращает массив с числовой индексацией;

MYSQL_BOTH – к полям возвращаемой записи можно обращаться как по числовым, так и по ассоциативным индексам.

Листинг 22 содержит очередной вариант кода листингов 3 и 4. На этот раз используется функция *mysql_fetch_array*(), возвращающая ассоциативный массив полей.

Листинг 22. Выборка данных функцией *mysql_fetch_array().*

```
<?
<? php
$sock=mysql_connect(«localhost»,«nobody»,«sasnobody») or die(«Could not connect to MySQL server!»);
mysql_select_db(«godar»,$sock) or die(«Could not select company database!»);
$SQL=«
  select TN,FIO,CLASS from nariyd_new.FIO
  Where TP=301 And TIP=1»;
```

```
$Result=mysql_query($SQL,$sock);
$i=0;
while($row = mysql_fetch_array($Result)) {
  Echo $i++;
  Print $row[«FIO»], «; », «<br>»;
}
mysql_free_result($result);
?>
```

Листинг 22 выдает тот же результат, что и листинги 20 и 21.

1.4.5.9. Mysql_free_result()

Функция *mysql_free_result()* освобождает память результата. Синтаксис функции:

int mysql_free_result(int result).

Функция *mysql_free_result()* должна быть использована только в том случае, если Вы беспокоитесь об использовании слишком большого объема памяти во время работы вашего скрипта. Вся используемая результатом память для определенного идентификатора результата автоматически будет освобождена.

1.4.5.10. Mysql_field_name

Функция *mysql_field_name* получает имя определенного поля в результате. Синтаксис функции:

string mysql_field_name(int result, int field_index).

Функция возвращает имя указанного поля. Аргументами функции являются идентификатор результата и индекс поля, т.е. *mysql_field_name*(*$result*,2). Возвратит имя второй области в результат, связанный с идентификатором результата(*$result*).

1.4.5.11. Mysql_field_table

Функция *mysql_field_table* получает имя таблицы, в которой находится указанное поле. Синтаксис функции:

string mysql_field_table(int result, int field_offset).

Функция получает имя таблицы поля.

1.4.5.12. Mysql_field_type

Функция *mysql_field_type* получает тип указанного поля в результате. Синтаксис функции:

string mysql_field_type(int result, int field_offset).

Функция подобна функции *mysql_field_name()*. Аргументы идентичны, но возвращается тип поля. Это будет что-то из «*int*», «*real*», «*string*», «*blob*», или другие типы, которые подробно описываются в документации *MySQL*. Пример в листинге 23:

Листинг 23

```
<?php
mysql_connect(«localhost:3306»);
mysql_select_db(«wisconsin»);
$result = mysql_query(«SELECT * FROM onek»);
$fields = mysql_num_fields($result);
$rows = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo «Your»«.$table.»«table has».$fields.«fields and».$rows «records <BR>»;
echo «The table has the following fields <BR>»;
while ($i < $fields) {
   $type  = mysql_field_type  ($result, $i);
   $name  = mysql_field_name  ($result, $i);
   $len   = mysql_field_len   ($result, $i);
   $flags = mysql_field_flags ($result, $i);
   echo $type.« ».$name.« ».$len.« ».$flags.«<BR>»;
   $i++;
}
mysql_close();
?>
```

1.4.5.13. Mysql_field_len

Функция *mysql_field_len* возвращает длину указанного поля. Синтаксис функции:

int mysql_field_len(int result, int field_offset).

1.4.5.14. Mysql_list_fields

Функция *mysql_list_fields* показывает список полей *MySQL* в результате запроса. Синтаксис функции:

int mysql_list_fields(string database_name, string table_name, int [link_identifier]).

Функция *mysql_list_fields()* извлекает информацию о заданной *tablename* таблице. Аргументы – имя базы данных и имя таблицы. После выполнения возвращается указатель результата, который может использоваться функциями *mysql_field_flags(), mysql_field_len(), mysql_field_name(),* и *mysql_field_type()*. Идентификатор результата является положительным целым. Функция возвращает -1, если происходит ошибка. Строка описания ошибки будет помещена в переменную $*phperrmsg*, и если функция не была вызвана как *@mysql(),* то затем также будет выведено это описание ошибки.

1.4.5.15. Mysql_num_fields

Функция *mysql_num_fields* получает количество полей в результате. Синтаксис функции:

int mysql_num_fields(int result).

Функция *mysql_num_fields()* получает количество полей в установленном результате.

1.4.6. Пример вывода таблицы в форму и результат выбора параметра

Пример формы обработки таблицы из базы данных приведен в листинге 24, результат листинга показан на рис. 1.38.

Листинг 24. Файл *test*15*_sql.php.*

```
<html>
   <head>
      <title>Пример</title>
   </head>
 <body>
<?php
$sock=mysql_connect(«localhost»,«nobody», «»);//соединение с сервером
mysql_select_db(«test»,$sock);// соединение с БД test
$SQL=«SELECT kod, name from test.syrnal»; // SQL запрос
$R=mysql_query($SQL,$sock);// обработка SQL-запроса
echo «<h1><FORM ACTION=«test16_sql.php» METHOD=POST>»; // форма передачи параметров
echo «Список журналов:»;
echo «<SELECT NAME=surnal[]>»; // объявление переменной
```

```
echo «<OPTION value=0>Все»;
if ($R) {
  $Tcol=mysql_NumRows($R); // количество записей в запросе
  if ($Tcol) {
        for ($i=0;$i<=$Tcol-1;$i++) { // обработка запроса по записям
              $p1=mysql_fetch_array($R);
              echo «<OPTION value=».$p1[kod].«>».$p1[name]; // вывод
                    каждой записи
        }
  }
}
echo «</SELECT>»;
echo «<INPUT TYPE=SUBMIT VALUE=« Вывод»>»; // вывод результата
echo «</FORM></h1>»;
$sock=mysql_close; // закрытие БД
?>
</body>
</html>
```

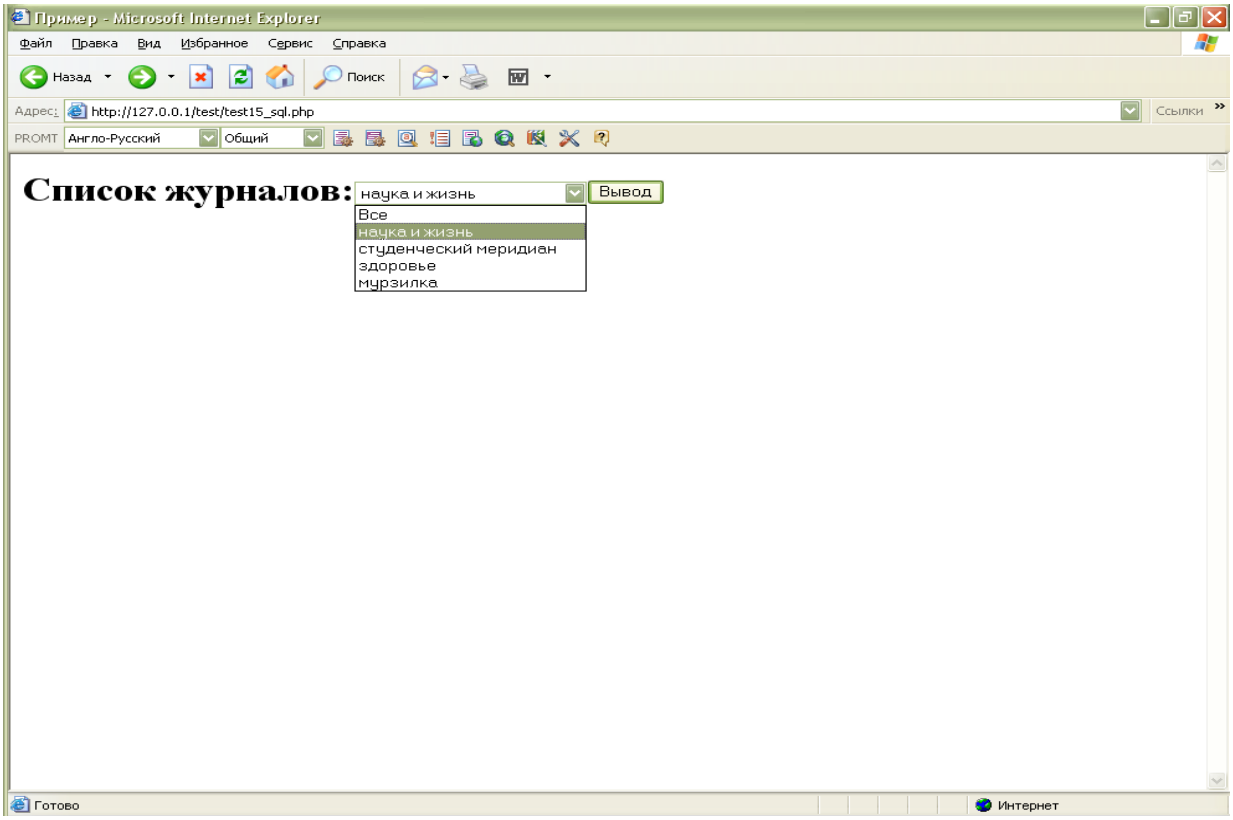


Рис. 1.38. Результат листинга 24

После нажатия кнопки «Вывод», выводится скрипт *test*16*_sql.php* (см. листинг 25 и рис. 1.39).

Листинг 25. Текст программы *test*16*_sql.php*.

```
<html>
   <head>
      <title>Список журналов</title>
   </head>
<body>
<?php
$sock=mysql_connect(«localhost»,«nobody»,«»); //соединение с сервером
mysql_select_db(«test»,$sock);// соединение с БД «test»
$sur=$_REQUEST[«surnal»][0]; // обработка переменной
$SQL=«SELECT kod, name from test.syrnal»; // SQL запрос из БД «test»
if ($sur<>0) {$SQL.= «where kod=».$sur;}
$R=mysql_query($SQL,$sock); // обработка SQL-запроса
echo «<center>»;
echo «Список журналов»,«<br><br>»; // заголок WEB -страницы
echo «
<TABLE BORDER=1 CELLPADDING=2 CELLSPACING=0>
  <TR>
  <TH>код</TH>
  <TH>название</TH>
  </TR>»; // шапка таблицы
if ($R) {
$Tcol=mysql_NumRows($R);
  if ($Tcol) {
  for ($i=0;$i<=$Tcol-1;$i++) {// обработка запроса по записям
        $p1=mysql_fetch_array($R);
        echo «<tr>»;
        echo «<TD ALIGN=RIGHT>»,$p1[0],«</TD>»; // вывод поля kod
        echo «<TD ALIGN=RIGHT>»,$p1[1] «</TD>»; // вывод поля name
        echo «</tr>»;
  }
```

```
  }
}
echo «</TABLE>»; // закрытие таблицы
$sock=mysql_close; // закрытие сервера
?>
</body>
</html>
```

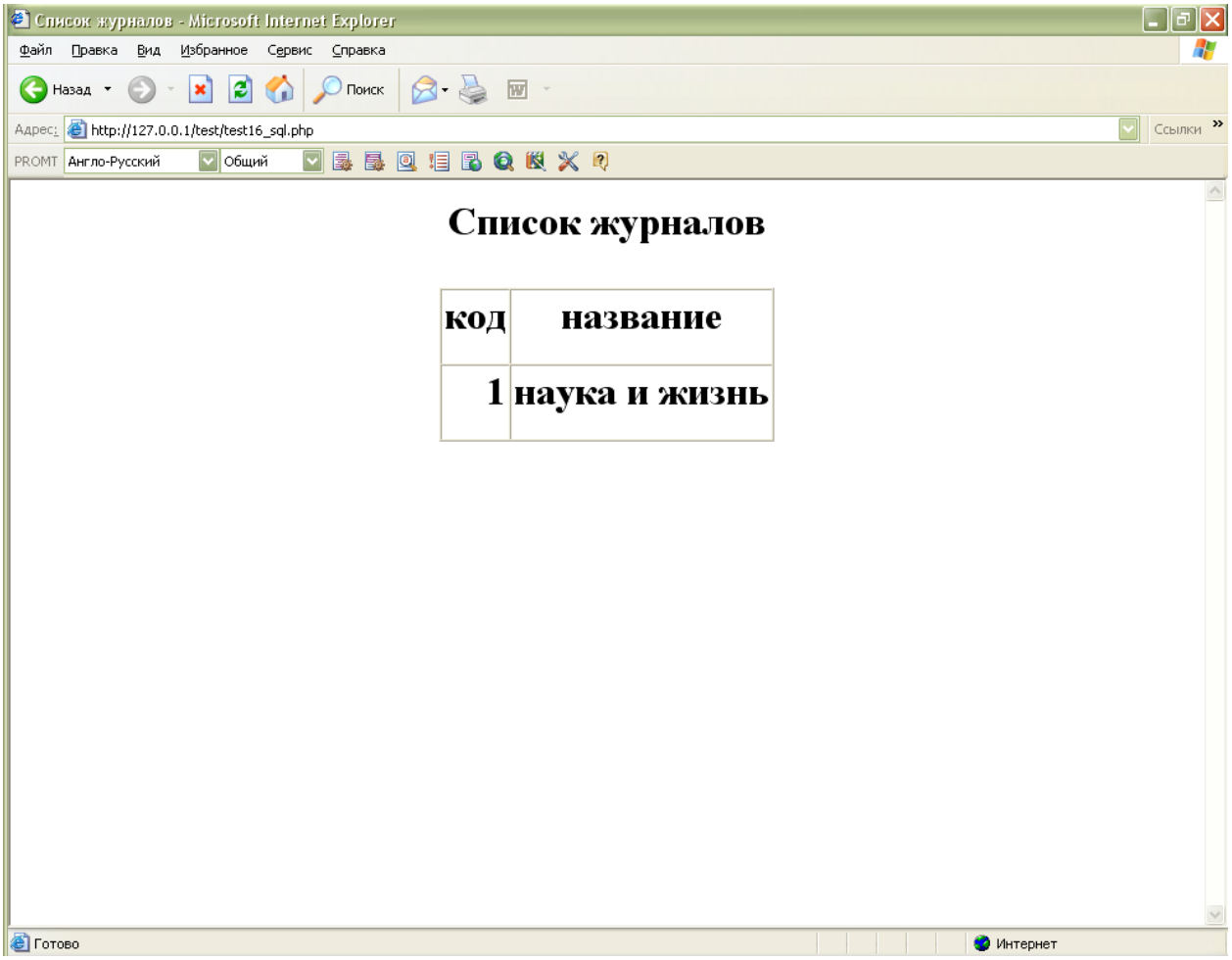


Рис. 1.39. Результат листинга 25

1.4.7. Задания для самостоятельной работы

1. Откройте пакет «*XAMPP Control Panel*». Ярлык на рабочем столе или в папке *C:/xampp/xampp-control.exe*. Запустите локальный *Web*-сервер *Apache* для *Windows* и *MySQL*.
2. Создайте базу данных под названием test (если она не создана). Создать таблицу с заданным количеством полей. Для выполнения задания внести 8 – 10 необходимых записей на усмотрение студента. Для создания таблицы можно использовать пакет «*MySQL-Front*». *Server* – «*localhost*»,

имя пользователя и пароль – пустые значения. Внешний вид пакета «*MySQL-Front*» представлен на рис. 1.40.

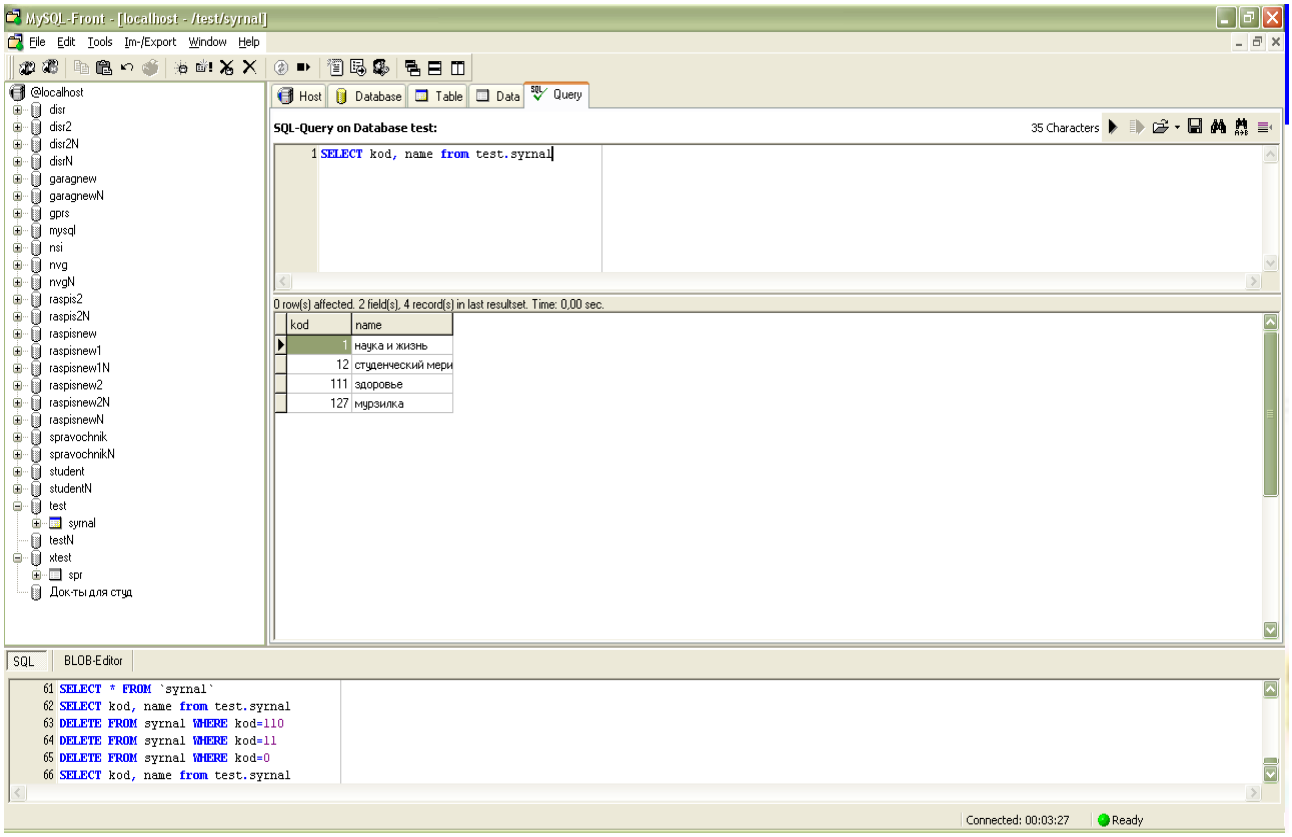


Рис. 1.40. Пример обработки результата запроса в приложении «*MySQL-Front*» в базе данных «*text*»

3. Используя пакет «*NotePad++*», создайте первый сайт с расширением **.php*. Файл сохраните в папке, например, *gruppa1* директории *C:/xampp/htdocs/*. Запустите его из адресной строки браузера, например:

   – по доменному адресу *«http://localhost/gruppa1/primer1.php »*

   – или по *IP*-адресу *«http://127.0.0.1/gruppa1/primer1.php»*.

   При отладке скрипта в браузере для обновления сайта используйте клавишу *F5* или кнопку «Обновить».

4. Создайте второй сайт из предложенного задания. Все данные из полей формы *PHP* помещайте в глобальный массив *$_REQUEST*.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК К ГЛАВЕ 1

1.1. Гофман В.Э. Delphi 5 / В.Э. Гофман, А.Д. Хомоненко. СПб.: БХВ Санкт-Петербург, 2000. 800 с.

1.2. Баас Р. Delphi 5 для пользователя: пер. с нем. / Р. Баас, М. Фервай, Х. Гюнтер: Киев: Издательская группа BHV, 2000. 496 с.

1.3. Бобровский С. Delphi 5: учеб. курс / С. Бобровский. СПб.; М.; Харьков; Минск: Питер, 2000. 638 с.

1.4. Фаронов В.В. Delphi 5: учеб. курс / В.В. Фаронов. М.: Нолидж, 2001. 608 с.

1.5. Архангельский А.Я. Delphi 5: справ. пособие / А.Я. Архангельский. М.: БИНОМ, 2001. 768 с.

1.6. Тюкачев Н. Delphi 5. Создание мультимедийных приложений: учеб. курс / Н. Тюкачев, Ю. Свиридов. СПб.; М.; Харьков; Минск: Питер, 2001. 400 с.

1.7. Баженова И. Ю. Delphi 5: самоучитель программиста / И.Ю. Баженова. М.: КУДИЦ-ОБРАЗ, 2000. 336 с.

1.8. DELPHI. Советы программистов. 2-е изд., доп. / под ред. В. Озерова. СПб.: Символ-Плюс, 2003. 976 с.

1.9. Котеров Д.В. PHP 5 / Д.В. Котеров, А.Ф. Костарев. СПб.: БХВ-Петербург, 2007. 1120 с.

1.10. Грофф Дж. Энциклопедия SQL. 3-е изд. (+ CD) / Дж. Грофф, П.Вайнберг. СПб.: Питер, 2003. 896 с.

1.11. Шкарина Л. Язык SQL: учебный курс / Л. Шкарина. СПб.: Питер, 2001. 592 с.

1.12. Яргер Р. MySQL и mSQL. Базы данных для небольших предприятий и Интернета / Р. Яргер, Дж. Риз, Т. Кинг. СПб.: Символ-Плюс, 2001. 560 с.

1.13. Сайт *http://www.mysql.com*

2. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ

2.1. Моделирование систем массового обслуживания с помощью GPSS

2.1.1. Основные принципы языка GPSS

GPSS (*General Purpose Simulating System* – общецелевая система моделирования) является языком имитационного моделирования, используемым для построения моделей и проведения моделирования на ЭВМ [2.1, 2.2, 2.3]. Модели на *GPSS* компактны, часто состоят из меньшего числа операторов, чем такие же модели, написанные на процедурных языках (например, на Паскале или Си). Это объясняется тем, что в *GPSS* встроено максимально возможное число логических программ, необходимых для моделирующих систем. В него также входят специальные средства для описания динамического поведения систем, меняющихся во времени, причем изменение состояний происходит в дискретные моменты времени. *GPSS* очень удобен при программировании, поскольку интерпретатор *GPSS* (здесь и далее интерпретатором называется моделирующая часть системы *GPSS*) многие функции выполняет автоматически. Например, *GPSS* без специального на то указания пользователя собирает статистические данные, описывающие поведение модели, автоматически печатает итоговую статистику по завершении моделирования. Пользователю нет необходимости включать в модель операторы для сбора и накопления этих данных или задавать формат, указывающий, в каком виде должны быть распечатаны итоговые данные. В язык включены и многие другие полезные элементы. Например, *GPSS* обслуживает таймер модельного времени, планирует события, которые должны произойти позднее в течение времени моделирования, вызывает их своевременное появление и управляет очередностью поступления.

Ниже будет представлена на процедурном уровне модель системы обслуживания с одним прибором и очередью. Сначала описан способ функционирования этой системы и поставлена цель разработать модель на ЭВМ, которая промоделировала бы данную систему. Далее рассмотрены основные вопросы, связанные с созданием такой модели. При моделировании такой системы предполагается, что существует генератор случайных чисел. Считают, что обращение к генератору происходит, как к функции, которая выдает значения случайных чисел, равномерно распределенных в интервале от 0,000000 до 0,999999 включительно.

2.1.1.1. Система обслуживания с одним прибором и очередью

Рассмотрим систему, состоящую из одного человека, выполняющего обслуживание определенного вида. Этот человек может быть кассиром, продающим билеты на станции, контролером в универсальном магазине, парикмахером в парикмахерской с единственным креслом. «Клиенты»

приходят к такому «обслуживающему прибору» в случайные моменты времени, ждут своей очереди на обслуживание (если есть необходимость), их обслуживают по принципу «первый пришел – первым обслужен». После этого они уходят. Схематично эта ситуация показана на рис. 2.1, где прямоугольник – это обслуживающий прибор, а кружок внутри него – заявка, находящаяся на обслуживании.

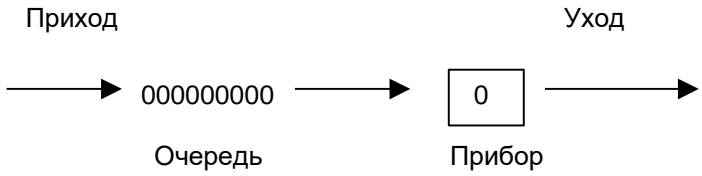


Рис. 2.1. Простая система массового обслуживания

2.1.1.2. Основные понятия, используемые при моделировании систем на GPSS

Для дальнейшего рассмотрения системы введем следующие определения:

СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ – это система, состоящая из обслуживающего прибора, заявки, находящейся на обслуживании, и ожидающих обслуживания заявок. Простая система массового обслуживания, изображенная на рис. 2.1, характеризуется двумя независимыми случайными переменными:

МОДЕЛЬНОЕ ВРЕМЯ – это промежуток времени между началом моделирования и его завершением.

ОЧЕРЕДЬ – это группа заявок, ожидающих обслуживания.

ИНТЕРВАЛ ПРИБЫТИЯ заявок – это интервал времени между последовательными моментами прибытия заявок в систему.

ВРЕМЯ ОБСЛУЖИВАНИЯ – это время, требуемое прибору для выполнения обслуживания.

СИСТЕМНАЯ ВЕЛИЧИНА – это величина, зависящая от значения первых двух независимых случайных переменных. Ниже перечислены некоторые из этих случайных величин, являющихся также случайными переменными:

1. Число заявок, прибывших на обслуживание за заданный промежуток времени.

2. Число заявок, которые попали на обслуживание сразу же по прибытии.

3. Среднее время пребывания заявок в очереди.

4. Средняя длина очереди.

5. Максимальная длина очереди.

6. Нагрузка прибора, являющаяся функцией времени, которое потрачено прибором на обслуживание в течение заданного промежутка времени.

Распределения этих системных величин и являются предметом исследования. Следует заметить, что разработку логической схемы модели на ЭВМ, которая будет имитировать систему обслуживания с одним прибором и очередью, нужно вести при следующих условиях:

1. Случайные переменные ИНТЕРВАЛ ПРИБЫТИЯ и ВРЕМЯ ОБСЛУЖИВАНИЯ являются равномерно распределенными и принимают только целые значения.

2. Все прибывающие заявки должны быть обслужены независимо от длины очереди.

3. Вначале моделирования система «пуста», т.е. нет очереди, и обслуживающий прибор свободен.

4. Моделирование продолжается до тех пор, пока не будет достигнуто значение модельного времени, заданное для этой модели в качестве одного из входных данных.

2.1.1.3. Элементы процедуры решения

При моделировании систем массового обслуживания совершаются некоторые СОБЫТИЯ. Все события в системе должны быть каким-либо образом зафиксированы, и должно быть учтено их воздействие на текущее состояние системы. Кроме того, необходимо определить, как нужно корректировать состояние системы в связи с воздействием на нее этих событий. События разделяются на две категории:

1. Основное событие – это такое событие, время возникновения которого можно запланировать заранее. Это, например, приход заявки, начало обслуживания и завершение обслуживания.

2. Вспомогательное событие – это событие, время возникновения которого невозможно запланировать заранее. Эти события возникают тогда же, когда и основные, но являются зависимыми, возникающими как следствие основных.

Так как работа модели связана с последовательным возникновением событий, то вполне естественно использовать понятие ТАЙМЕР МОДЕЛЬНОГО ВРЕМЕНИ в качестве одного из элементов модели системы. Для этого вводят специальную переменную и используют ее для фиксации текущего времени работы модели. Опишем теперь некоторые специфические свойства таймера модельного времени.

Когда начинается моделирование, таймер модельного времени обычно устанавливают на нулевое значение. Разработчик сам решает вопрос о том, какое значение реального времени принять за точку отсчета. Например, началу отсчета может соответствовать 8 ч утра первого моделируемого дня. Разработчик также должен решить вопрос о выборе величины единицы

времени. Единицей времени может быть 1 с, 5 с, 1 мин, 20 мин или 1 ч. Когда единица времени выбрана, все значения времени, получаемые при моделировании или входящие в модель, должны быть выражены через эту единицу.

На практике значения модельного времени должны быть достаточно малыми по сравнению с реальными промежутками времени, протекающими в моделируемой системе. В данной системе обычно выбирают единицу времени, равную 1 мин. Если при моделировании некоторой системы при текущем значении модельного времени ее состояние изменилось, то нужно увеличить значение таймера. Чтобы определить, на какую величину должно быть увеличено значение таймера, используют один из двух методов:

1. Концепция фиксированного приращения значений таймера.

При таком подходе увеличивают значение таймера ровно на одну единицу времени. Затем нужно проверить состояние системы и определить те из запланированных событий, которые должны произойти при новом значении таймера. Если таковые имеются, то необходимо выполнить операции, реализующие соответствующие события, снова изменить значение таймера на одну единицу времени и т.д. Если проверка покажет, что для нового значения таймера не запланировано ни одного события, то произойдет передвижение таймера непосредственно к следующему значению.

2. Концепция переменного приращения значений таймера.

В этом случае условием, вызывающим приращение таймера, является наступление времени «близкого события». Близкое событие – это то событие, возникновение которого запланировано на момент времени, равный следующему ближайшему значению таймера модельного времени. Колебания приращения таймера от случая к случаю объясняют выражение «переменное приращение времени».

Следует заметить, что выгоднее использовать концепцию переменного приращения значений таймера, т.к. при нем можно избежать обработки в промежуточные моменты времени, когда не планируется выполнение никаких событий. В рассматриваемой задаче с одним обслуживающим прибором и очередью будет использована именно эта концепция. И один из способов определения времени ближайшего события заключается в составлении списка моментов времени, в которые запланированы выполнения различных основных событий. Когда приходит время наращивания значения таймера, в этом списке должно быть найдено ближайшее событие. (Если список моментов времени отсортирован в порядке возрастания, то никакого поиска не требуется. Таймер просто установится в нужное значение в соответствии с первой позицией списка). Затем таймеру должно быть придано его значение, и управление должно быть передано в ту часть модели, где обслуживается событие, о котором идет речь.

Обычно после какого-то момента времени наступает необходимость прекратить моделирование. Например, нужно предотвратить приход новых заявок в систему, но обслуживание надо продолжать до освобождения системы. Одним из способов является введение в модель основного псевдособытия, называемого ЗАВЕРШЕНИЕМ МОДЕЛИРОВАНИЯ. Тогда одной из функций модели будет планирование этого события. Момент времени, наступление которого должно вызвать остановку моделирования, задается обычно в виде числа. Таким образом, в процессе моделирования нужно проверять, является ли событие «завершение моделирования» следующим событием. Если «да», то в таймере устанавливается значение времени конца моделирования, а управление передается процедуре, которая отрабатывает завершение моделирования.

Мы рассмотрели различные элементы, из которых формируется модель системы обслуживания с одним прибором и очередью. Последним шагом является сбор этих элементов в единую общую модель. В общем виде логическую схему модели можно представить в виде блок-схемы на рис. 2.2.

Теперь наша задача состоит в создании машинной модели на ЭВМ, которая позволит изучить поведение системы в течение времени моделирования. Иначе говоря, нужно реализовать эту блок-схему на ЭВМ, используя блоки и операторы языка GPSS.

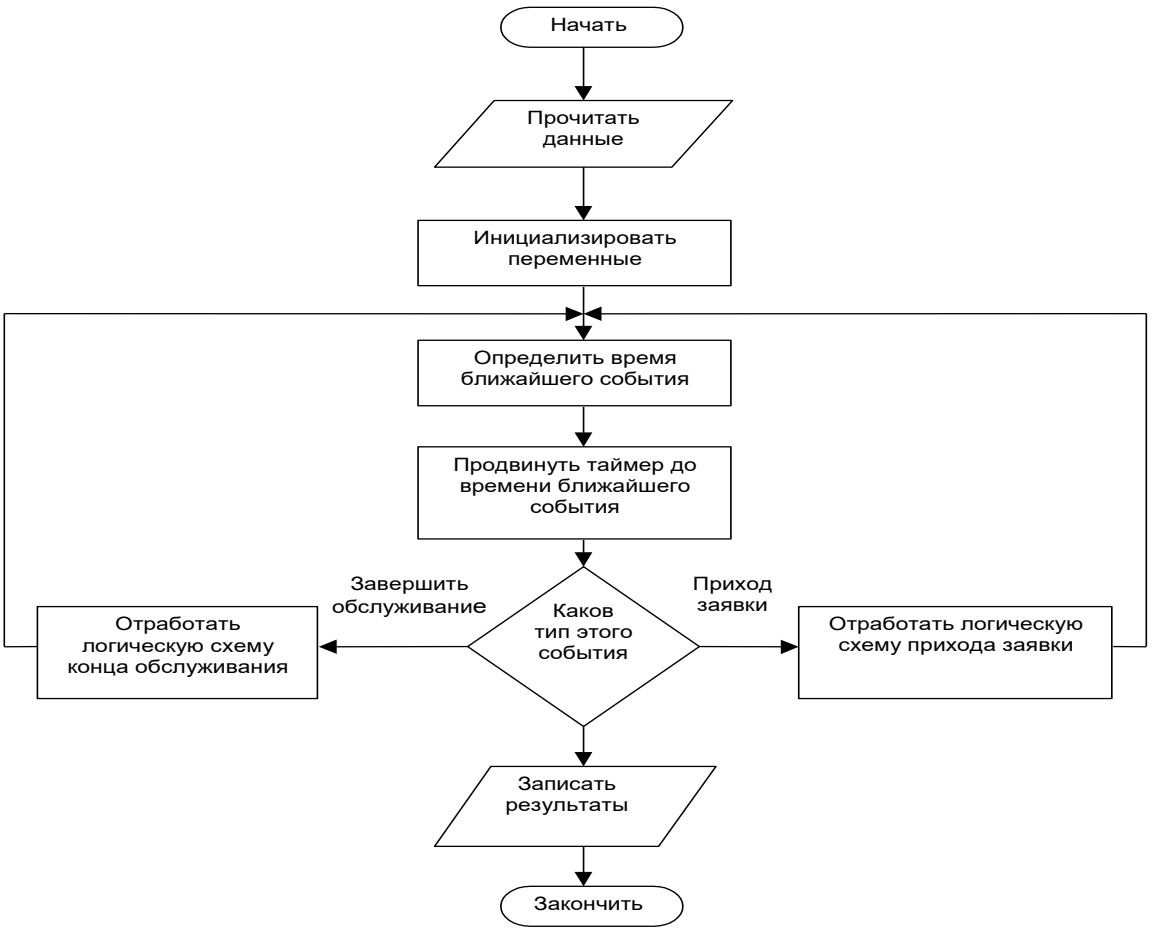


Рис. 2.2. Блок-схема GPSS-модели

2.1.2. Основные концепции моделирования на GPSS

GPSS представляет собой язык и машинную программу. Как любой язык, он содержит словарь и грамматику, с помощью которых легко могут быть разработаны точные модели систем определенного типа. Машинная программа интерпретирует модель, написанную на языке *GPSS*, предоставляя тем самым пользователю возможность проведения экспериментов с этой моделью на ЭВМ. Эта машинная программа и называется интерпретатором.

В этом разделе будут рассмотрены свойства основного подмножества блоков *GPSS*. Это подмножество выбрано таким образом, чтобы можно было создавать законченные, относительно простые модели систем на GPSS.

2.1.2.1. Динамические элементы моделей GPSS (транзакты)

Для того чтобы показать пошаговое выполнение процедур, было использовано символическое изображение передач управления в виде специальных фигур и линий (см. рис. 2.2). Управление передавалось от одного блока к другому (или от одного действия к другому). Однако в *GPSS* концепция «передачи управления от блока к блоку» имеет специфические особенности и требует более подробного рассмотрения. Конфигурация блок-схемы *GPSS*-модели отражает направления, по которым происходит движение перемещающихся элементов. Каждый такой элемент называется ТРАНЗАКТОМ. Транзакты являются динамическими (т.е. движущимися) элементами GPSS-модели. Работа этой модели заключается в перемещении транзактов от блоков к блокам. Некоторые примеры возможных аналогий между транзактами и элементами реальных систем представлены в табл. 2.1.

Таблица 2.1

Аналогии между элементами реальных систем и транзактами

Системы	Элементы систем, символизируемые транзактами
Большой универсальный магазин	Покупатель
Автомобильное шоссе	Автомобиль
Радиомастерская	Радиоприемник
Склад	Заявка
Парикмахерская	Клиент

Таким образом, перемещение транзакта от блока к блоку в модели аналогично, например, передвижению клиента в парикмахерской от одной стадии к другой.

В самом начале моделирования в GPSS-модели нет ни одного транзакта. В процессе моделирования транзакты входят в модель в определенные моменты времени в соответствии с теми логическими потребностями, которые возникают в моделируемой системе. Подобным же образом транзакты

покидают модель. В общем случае в модели существует большое число транзактов, но в один момент времени двигается только один.

Если транзакт начал свое движение, он перемещается от блока к блоку по пути, предписанному блок-схемой. Такое продвижение транзакта продолжается до тех пор, пока не произойдет одно из следующих возможных событий:

1) транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное время;

2) транзакт входит в блок, функцией которого является удаление транзакта из модели;

3) транзакт «пытается» войти в следующий блок в соответствии с блок-схемой, но блок «отказывается» принять его.

Если возникло одно из описанных условий, то транзакт остается на месте и начинается перемещение в модели другого транзакта. Таким образом, выполнение моделирования в системе продолжается.

Модель на GPSS состоит из одного или нескольких независимых сегментов. В процессе моделирования активным является тот из сегментов, в котором находится перемещающийся в настоящий момент транзакт. Когда он блокируется, начинает двигаться следующий транзакт, и может быть так, что этот следующий транзакт принадлежит другому сегменту модели. Таким образом, происходит переключение активности между сегментами.

2.1.2.2. Моделирование многоканальных устройств

Прибор в GPSS используют для моделирования единственного устройства обслуживания. Два или более находящихся рядом обслуживающих устройства могут быть промоделированы на GPSS двумя или более приборами, располагаемыми рядом, т.е. параллельно. Этими приборами могут быть, например, два парикмахера, работающие рядом.

Для моделирования однородных (обладающих определенными общими свойствами) параллельных приборов GPSS предоставляет специальное средство (или элемент).

Для этого используют название «многоканальное устройство» (МКУ). Схематично такая ситуация представлена на рис. 2.3. В модели может быть несколько многоканальных устройств. Для того чтобы между ними было различие, им можно давать имена. Условия использования имен такие же, как и в случае приборов и очередей. Число приборов, которое моделируется каждым из многоканальных устройств, определяется разработчиком. Здесь употребляют термин «емкость многоканального устройства».

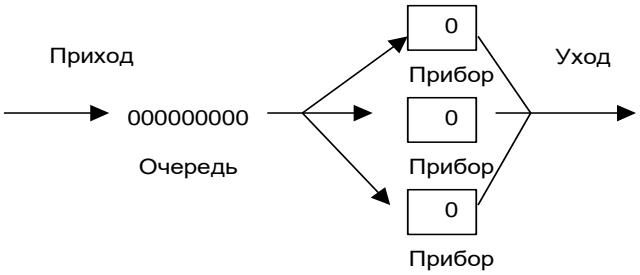


Рис. 2.3. Три параллельно работающих прибора

Элементом, который занимает и использует устройство, является транзакт. При этом происходят следующие события: 1) транзакт ожидает своей очереди (если необходимо); 2) транзакт занимает устройство; 3) устройство осуществляет обслуживание; 4) транзакт освобождает устройство.

2.1.3. Основные блоки языка GPSS

Все блоки записываются с первой позиции строки, сначала идет имя блока, а затем, через запятую, параметры. В записи параметров не должно быть пробелов. Если какой-то параметр в блоке отсутствует (задается по умолчанию), то соответствующая ему запятая остается (если это не последний параметр). Если в первой позиции строки стоит символ *, то эта строка с комментарием.

1. GENERATE A,B,C,D,E,F

Создает транзакты через определенные интервалы времени.

A – средний интервал времени между появлениями транзактов.

B – 1) если число, то это половина поля, в котором равномерно распределено значение интервала между появлениями транзактов [A-B, A+B];

– 2) если функция, то для определения интервала значение A умножается на значение функции.

C – момент времени появления первого транзакта.

D – предельное количество транзактов.

E – величина приоритета транзакта.

F – число параметров у транзакта и их тип (PB – байтовый целый, PH – полусловный целый, PF – полнословный целый, PL – с плавающей запятой, где P – число параметров у транзакта).

2. TERMINATE A

Уничтожает транзакты из модели и уменьшает значение счетчика завершения на A единиц. Работа модели завершится, если счетчик завершения станет меньше или равен нулю. Если параметр A отсутствует, то блок просто уничтожает транзакты.

Примеры использования блока GENERATE:

1) GENERATE 5,3 – блок генерирует транзакты через 5±3 единицы системного времени, т.е. интервалом прибытия является случайное число со средним значением, равным 5, и полем допуска, равным 6.

2) GENERATE 10 – генерируется поступление транзактов в систему через каждые 10 единиц системного времени.

3) GENERATE 3,3,10,5 – моментом прихода первого транзакта является значение 10. После этого интервалы времени прибытия находят из равномерного распределения 3±3. Однако только первые пять транзактов должны войти в модель через этот блок.

4) GENERATE 5,,,,,2h – транзакты, имеющие два полусловных целых параметра, поступают в систему каждые 5 единиц системного времени.

3. SEIZE A

Если прибор с именем A свободен, то транзакт занимает его (переводит в состояние «занято»), если нет, то ставится в очередь к нему. Именем прибора может быть числовой номер или последовательность от 3 до 5 символов.

4. RELEASE A

Транзакт освобождает прибор с именем A, т.е. переводит его в состояние «свободно».

5. ADVANCE A,B

Задерживает обработку транзакта данным процессом и планирует время начала следующего этапа обработки.

A – среднее время задержки.

B – имеет тот же смысл, что и для GENERATE.

Пример 1

Рассмотрим систему, состоящую из кассы и очереди к ней. Клиент обслуживается в кассе за время (3±1) мин. Очередь пополняется за счет вновь прибывших через (5±4) мин. Программа будет выглядеть следующим образом:

SIMULATE

GENERATE 5,4	Моделирование прихода клиентов в систему через (5±4) мин
SEIZE KAS	Если прибор с именем KAS находится в состоянии «свободно», то транзакт занимает его (клиент обслуживается в кассе), в противном случае встает в очередь
ADVANCE 3,1	Клиент обслуживается в кассе (3±1) мин
RELEASE KAS	Обслуживание клиента в кассе закончено (транзакт

освобождает прибор KAS, переводя его в состояние «свободно»

TERMINATE 1 Клиент покидает систему (транзакт уничтожается и счетчик завершения уменьшается на 1)

START 100

END

В данной программе, кроме описанных выше, использовались еще 3 блока:

SIMULATE, START и END.

Карта SIMULATE всегда ставится в начале программы. Если она отсутствует, то моделирование производиться не будет.

Карта START имеет следующий формат:

START A,B,C,D

A – начальное значение счетчика числа завершений.

B – признак подавления печати. Если задан параметр NP, то стандартная печать в конце моделирования производиться не будет.

C – число завершений, через которые будет выдаваться промежуточная печать.

D – признак печати цепей. Печать цепей производится, если на месте D стоит 1.

Карта END должна быть последней картой программы.

6. QUEUE A

Собирает статистику о входе транзакта в очередь с именем A.

Именем очереди может быть числовой номер или последовательность от 3 до 5 символов.

7. DEPART A

Собирает статистику о выходе транзакта из очереди с именем A.

Пример 2

Если в задаче из предыдущего примера необходима информация об очереди, то программу нужно дополнить.

```
SIMULATE
GENERATE 5,4
QUEUE QQQ
SEIZE KAS
DEPART QQQ
ADVANCE 3,1
```

RELEASE KAS

TERMINATE 1

START 100

END

В данном примере группа операторов QUEUE-DEPART позволяет дать имя очереди к прибору и собирает статистику об этой очереди.

8. ENTER A,B

Если в многоканальном устройстве с именем A имеются свободные каналы, то транзакт занимает B каналов (по умолчанию 1). Если свободных каналов нет, то транзакт становится в очередь и его обработка процессом приостанавливается до освобождения каналов. Именем устройства может быть числовой номер или последовательность от 3 до 5 символов.

9. LEAVE A,B

Транзакт освобождает B каналов в многоканальном устройстве A.

Прежде чем использовать многоканальное устройство, его нужно описать картой STORAGE.

A STORAGE B

Здесь A – имя многоканального устройства, B – максимальная емкость.

Пример 3

Пусть заявки поступают на обработку с ограниченным числом мест в очереди, равным 3. Если очередь заполнена, то заявки покидают систему.

SIMULATE

LIM STORAGE 3

GENERATE 5,4

ENTER LIM

SEIZE PRB

LEAVE LIM

ADVANCE 3,2

RELEASE PRB

TERMINATE 1

START 100

END

Пример 4

Рассмотрим работу ЗУ емкостью 256 кБ. В ЗУ с интервалом 3±2 с загружаются файлы, занимающие 4 кБ памяти. Через 10±3 с файлы становятся ненужными и освобождают память.

```
    SIMULATE
MEM  STORAGE 256
    GENERATE 3,2
    ENTER MEM,4
    ADVANCE 10,3
    LEAVE MEM,4
    TERMINATE 1
    START 250
    END
```

10. ASSIGN A,B,C

Присваивает (если A+ – увеличивает, A- – уменьшает) параметру транзакта (номер которого указан в A) значение параметра B. C – тип параметра транзакта.

Параметр транзакта – это величина, соответствующая атрибуту элемента модели. У каждого транзакта свои параметры. Ссылка на параметр транзакта, который обрабатывается блоком, выполняется в виде группового имени P, за которым следует последовательный номер параметра и его тип. Например, P3H – третий полусловный параметр обрабатываемого транзакта.

Примеры использования блока ASSIGN:

ASSIGN 3,2,PH – третьему полусловному параметру присвоить 2.

ASSIGN 1-,1,PB – значение первого байтового параметра уменьшить на единицу.

11. SAVEVALUE A,B,C

Присваивает (если A+ – увеличивает, A- – уменьшает) сохраняемой величине, указанной в параметре A (только номер или последовательность от 3 до 5 символов, не начинающиеся на X), значение параметра B.

C – тип сохраняемой величины.

Ссылка на сохраняемую величину выполняется в виде группового имени X, за которым следует либо номер сохраняемой величины, либо после символа $ – символьное имя этой величины, например X$DASP.

Для задания начальных значений сохраняемым переменным используется карта INITIAL.

INITIAL A_1,B_1/ ... /A_N,B_N

A_i – имя сохраняемой величины,

B_i – ее начальное значение.

Если сохраняемая величина не описана в карте INITIAL, то по умолчанию ее значение равняется 0.

Примеры использования оператора SAVEVALUE:

SAVEVALUE SI,12,PF – присвоить переменной с именем SI полнословного целого типа значение 12.

SAVEVALUE 12,-3,PB – присвоить переменной байтового типа с номером 12 значение -3.

SAVEVALUE A+,1,PB – увеличить значение A на единицу.

12. TEST X A,B,C

Проверяет отношение X (L(<), LE(<или=), E(=), NE(<>), GE(>или=), G(>)) между значениями параметров A и B. Если оно выполняется, то транзакт обрабатывается следующим блоком, если нет, то блоком, на который указывает метка C. Именем метки может быть числовой номер или последовательность от 3 до 5 символов.

Примеры:

TEST NE 12,X$TAST,TST – управление передается блоку с меткой TST, если переменная TAST равна 12.

TEST LE P1H,P2H,GO – если второй параметр транзакта больше первого, то управление передается по метке GO.

Другой формой оператора TEST является оператор IF.

IF X,A

Осуществляет переход на метку A, если логическое условие X истинно. Если логическое условие X ложно, то транзакт обрабатывается следующим блоком.

Примеры использования оператора IF вместо TEST:

IF X$TAST=12,TST

IF P1H>P2H,GO

Пример 5

Пусть в данном примере транзакты соответствуют грузовым автомобилям. Грузоподъемность каждого автомобиля будет содержаться в параметре транзакта. Существует 2 вида грузовиков – грузоподъемностью 5 и 10 т. Они перевозят грузы на некоторый объект. Пусть в данный момент разгружаться может только один автомобиль. Поэтому на разгрузку образуется очередь,

которая будет иметь имя QQQ. Сам объект назовем OBJ. Масса груза, уже перевезенная на объект, будет храниться в ячейке MAS.

```
SIMULATE
INITIAL MAS,100          пусть на объект уже перевезено 100 т груза
; 1-й сегмент программы
; описывает поток автомобилей грузоподъемностью 5 т
GENERATE 15,5,,,,1B      автомобили подъезжают к объекту через 15±5 мин
ASSIGN 1,5,PB
QUEUE QQQ
SEIZE OBJ
DEPART QQQ
ADVANCE 5,1              задержка на разгрузку
SAVEVALUE MAS+,P1H,XH    увеличение значения массы груза на объекте
RELEASE OBJ
TERMINATE
; 2-й сегмент программы
; описывает поток автомобилей грузоподъемностью 10 т
GENERATE 25,10,,,,1B
ASSIGN 1,10,PB
QUEUE QQQ
SEIZE OBJ
DEPART QQQ
ADVANCE 10,2
SAVEVALUE MAS+,,P1H,XH
RELEASE OBJ
TERMINATE 1
START 250
END
```

13. TRANSFER A

Осуществляется безусловный переход на метку A. Другая форма:

GOTO A

14. PREEMPT A,B,C,D,E

Транзакт занимает прибор с именем A, т.е. если другой транзакт обслуживается в нем, то его обслуживание прерывается, и он передается на обработку блоку, указанному в C, причем в его параметре с номером D записывается время, оставшееся до конца обслуживания. Если B=PR, то обслуживание этого транзакта также может быть прервано транзактом с более высоким приоритетом. Если B опущено, то обслуживание этого транзакта прервать нельзя. Если E=RE, то прерванный транзакт теряет право на автоматическое восстановление в приборе.

15. RETURN A

Транзакт, захвативший прибор с именем A, освобождает его.

В GPSS можно обращаться к функциям, которые описываются с помощью карты FUNCTION. Существует 2 вида функций: дискретные (тип D) и непрерывные (тип C). Описание функций осуществляется следующим образом:

A FUNCTION B,CN

x_1,y_1/x_2,y_2/.../x_N,y_N

A – имя функции, B – аргумент функции, C – тип функции (C или D), N – количество точек описания функции. Именем функции может быть последовательность от 3 до 5 символов.

Обращение к функциям осуществляется с помощью группового имени FN, за которым после символа $ следует имя функции.

Пример использования функции:

FFF FUNCTION X$AS,D5

0,1/.2,2/.5,3/.8,4/1.2,5

Здесь FFF – имя функции, X$AS – аргумент, D5 означает, что функция дискретная и она описывается пятью точками, значения функции записываются без пробелов. Если AS=0.8, то значение функции равно 4 (FN$FFF = 4). Если количество знаков, описывающих функцию, больше 71, то можно перенести данные на следующую строку, опуская спецсимвол «/» между дискретными точками k и k+1.

В GPSS имеется восемь подпрограмм, которые вычисляют случайные числа, равномерно распределенные в интервале [0, 1]. Обращение к этим функциям выполняется с помощью группового имени RN, после которого следует номер генератора (от 1 до 8). Все генераторы совершенно одинаковы.

В языке GPSS используются стандартные числовые атрибуты, которые являются переменными модели, описывающими блоки, приборы, многоканальные устройства и очереди, значения которых вычисляет интерпретатор, а разработчик в описании модели их только использует. Каждый стандартный числовой атрибут имеет групповое имя, которое при использовании дополняется номером, или после символа $ именем этого

элемента модели (объекта), к которому он относится. Например, стандартный числовой атрибут Q1 содержит число транзактов в очереди с номером 1, R$MEM показывает остающуюся свободную емкость многоканального устройства MEM. Список групповых имен основных стандартных числовых атрибутов представлен в табл. 2.2.

Таблица 2.2

Объект	Групповое имя	Название
Блоки	N	Счетчик входов
	W	Счетчик текущего содержимого
Приборы	F	Состояние прибора (1 – занят, 0 – свободен)
	FC	Счетчик числа занятий
	FR	Коэффициент использования
	FT	Среднее время задержки на одно занятие
Многока-нальные устройства	R	Остающаяся свободной емкость
	S	Текущее содержимое
	SA	Среднее содержимое
	SC	Счетчик числа входов
	SR	Коэффициент использования
	SM	Максимальное содержимое
	ST	Среднее время задержки на единицу
Очереди	Q	Текущее содержимое
	QA	Среднее содержимое
	QC	Счетчик числа входов
	QM	Максимальное содержимое
	QT	Среднее время прерывания (на основе QC)
	QX	Среднее время пребывания (на основе QZ)
	QZ	Счетчик числа нулевых входов
Время	C1	Значение системного времени
Переменная	X	Значение переменной
Функция	FN	Вызов функции

Пример 6

Пусть процессор Пр контролирует процесс в каком-либо объекте по двум независимым друг от друга параметрам (см. рис. 2.4). Значения параметров от датчиков Д1 и Д2 поступают на анализирующее устройство (АУ), которое

сравнивает эти значения с оптимальными для данной системы и передает результаты сравнения в блок управления (БУ). БУ формирует сигналы, под воздействием которых происходит управление объектом с помощью исполнительных механизмов (ИМ1 и ИМ2). АУ является быстродействующим прибором, поэтому время задержки в нём можно принять равным нулю. Первый параметр анализируется каждые 2 с. Его нормальное значение составляет 20 единиц. В случае, если значение этого параметра отклоняется от нормального, БУ формирует сигналы увеличить (или уменьшить – в зависимости от ситуации) значение первого параметра. Значения второго параметра анализируются каждые 10 с, нормальное значение для этого параметра 40 единиц. Ситуация, когда значение второго параметра выйдет за пределы [30, 60] единиц, считается экстремальной и обрабатывается в первую очередь. Значение первого параметра хранит переменная с номером 1, значение второго – с номером 2. Случайные значения первого моделируются с помощью функции CHN, второго – с помощью функции DIF.

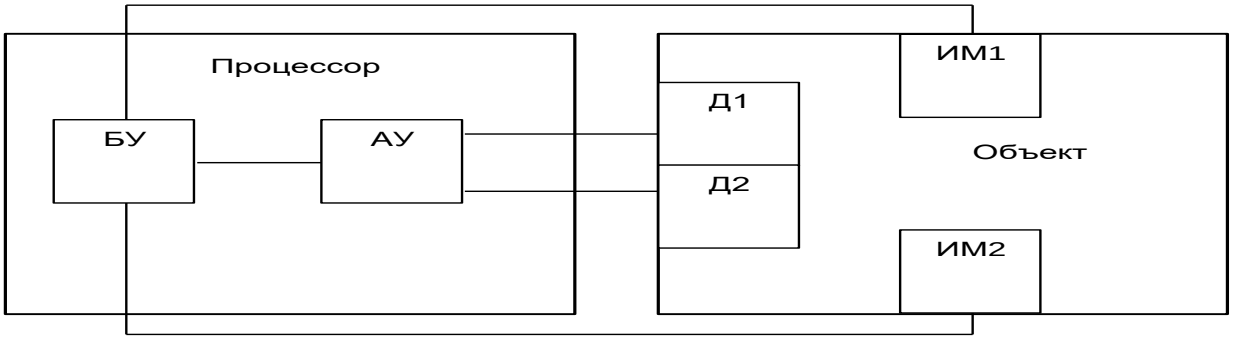


Рис. 2.4. Управление объекта процессором

```
    SIMULATE
    INITIAL 1,20/2,40          установка нормальных значений параметров
CHN FUNCTION RN2,D10
    0,0/.1,0/.2,1/.3,1/.4,2/.5,3/.6,4/.7,5/.8,8/1,8
DIF FUNCTION RN3,D5
    0.1,2/0.2,3/0.3,5/0.6,8/1,2
    GENERATE 2,,,,,1B          опрос значений первого параметра через 2 с
    IF RN1>0.5,TWO             значение параметра с равной вероятностью
                               может изменяться в ту или иную сторону, по
                               метке TWO обрабатывается ситуация, когда
                               значение параметра уменьшается
    SAVEVALUE 1+,FN$CHN,XB
    GOTO DDD1
TWO SAVEVALUE 1-,FN$CHN,XB
```

```
DDD1 ASSIGN 1,X1B,PB      параметр транзакта несет информацию о
                          значении первого параметра
     QUEUE QQQ
     SEIZE PROC           сообщение начинает обрабатываться
                          процессором
     DEPART QQQ
     ADVANCE 5,2
     IF P1B=20,DDD2       если значение параметра нормальное, то
                          никаких действий не нужно
     IF P1B>20,BBB1
     SAVEVALUE 1+,2,XB    увеличиваем значение параметра, если оно
                          было ниже нормального
     GOTO DDD2
BBB1 SAVEVALUE 1-,2,XB
DDD2 RELEASE PROC
     TERMINATE
     GENERATE 10,,,,,1B   опрос значений 2-го параметра через 10 с
     IF RN1>0.5,TWO2
     SAVEVALUE
2+,FN$DIF
     GOTO DDD3
TWO2 SAVEVALUE 2-,FN$DIF,XB
DDD3 ASSIGN 1,X2B,PB
     IF X2>60,ACD1        передача управления по ACD1 в случае
                          экстремальной ситуации
     IF X2<30,ACD2        передача управления по ACD2
     QUEUE QQQ
     SEIZE PROC
     DEPART QQQ
     ADVANCE 10,3
     IF P1B=40,DDD4
     IF P1B>40,BBB2
     SAVEVALUE 2+,3,XB
     GOTO DDD4
```

```
BBB2 SAVEVALUE 2-,3,XB
DDD4 RELEASE PROC
     GOTO KON
ACD1 PREEMPT PROC
     ADVANCE 15,1
     SAVEVALUE 2-,10,XB
     RETURN PROC
     GOTO KON
ACD2 PREEMPT PROC
     ADVANCE 16,2
     SAVEVALUE 2+,7,XB
     RETURN PROC
KON TERMINATE 1
     START 200
     END
```

16. SELECT X A,B,C,D

Среди стандартных атрибутов модели с групповым именем D, номер которого лежит в интервале от B до C, ищется наименьшее (X=MIN) и наибольшее (X=MAX), и номер этого атрибута записывается в параметр транзакта с номером, заданным в A.

Пример:

SELECT MIN 1,1,6,Q

Среди очередей с номерами от 1 до 6 ищется та очередь, которая имеет наименьшее текущее содержимое.

17. TABULATE A

В таблицу с именем A записывается значение указанной в ней переменной модели. Таблица должна описываться картой TABLE.

A TABLE B,C,D,E

A – имя таблицы,

B – имя аргумента,

C – левая граница,

D – ширина интервала,

E – количество интервалов.

2.1.4. Выходные данные. Значения выходных параметров

По завершении моделирования интерпретатор GPSS автоматически распечатывает некоторое количество информации о поведении модели. Эта информация включает статистические данные по каждому из элементов, используемых в модели, т.е. по каждому из приборов, очередей и т.д. Для того чтобы иметь представление о работе модели на основании результатов моделирования, ниже приведено описание *выходных параметров*.

1. Относительное время и абсолютное время – момент модельного времени, в котором завершено моделирование.

2. Прибор (словарь символов для приборов) – здесь указывается перечень приборов. В перечне представлены все символические имена приборов, используемые в модели.

3. Средняя нагрузка – время, в течение которого прибор был занят.

4. Число входов – число обслуживаний.

5. Среднее время/тран – среднее время интервала обслуживания.

6. Номер обслуживаемого транзакта – транзакт с таким-то номером находится на обслуживании в таком-то приборе в момент завершения моделирования.

7. Номер захватившего транзакта – номер транзакта, который захватил прибор при завершении моделирования.

8. Содержимое SAVEVALUE – содержимое значений параметров.

Выходные данные очереди:

1. Очередь (словарь символов для очередей).

2. Максимальное содержимое – наибольшее значение содержимого очереди, зарегистрированное в течение моделирования.

3. Среднее значение – среднее значение содержимого очереди.

4. Всего входов – общее число заявок, вошедших в очередь.

5. Ноль входов – общее число заявок, которые не стояли в очереди.

6. Процент нулей – процент заявок, которые не стояли в очереди.

7. Среднее время/тран – среднее время, проведенное в очереди с учетом нулевых входов.

8. $среднее время/транз – среднее время нахождения в очереди на один нулевой вход.

9. Текущее содержимое – текущее значение содержимого очереди.

10. Имя таблицы – имя таблицы, в которой представлено распределение времени ожидания для этой очереди.

Выходные данные многоканальных устройств:

1. Многоканальное устройство – словарь символов для многоканальных устройств.

2. Емкость – максимальная емкость.

3. Средняя нагрузка – время, в течение которого прибор был занят.

4. Входы – общее число занятий приборов.

5. Среднее время/тран – среднее время интервала обслуживания.

6. Максимальное содержимое – максимальное число занятий прибора.

7. Среднее содержимое – среднее число занятий прибора.

8. Текущее содержимое – занятая емкость на момент завершения.

Выходная статистика по примеру 6:

Устройство	Средняя загрузка	Число входов	Среднее время/транз.
PROC	.80	99	8.0

Очередь	Max значение	Среднее значение	Всего входов	Ноль входов	Процент нулей
QQQ	3	1.20	99	10	10.10

Среднее время/транз.	$среднее время/транз.	Текущее значение
4.87	4.90	3

Содержимое SAVEVALUE:

A1 = 17.00

A2 = 8.00

2.1.5. Запуск интерпретатора GPSS

Файл с программой можно создать в любом текстовом редакторе, но он должен быть сохранен в формате «текст DOS», например в Блокноте с расширением .txt. Для проведения полного факторного эксперимента лучше, если имя файла будет содержать одну цифру или букву. Для запуска программы на языке GPSS необходимо запустить файл GPSS.EXE и указать интерпретатору полное имя файла, в котором находится листинг программы, находящийся в этом же каталоге, например 1.txt.

Для сохранения выходных данных в файле необходимо в командной строке указать путь к этому файлу: gpssr.exe > 2.txt. После этого ввести имя файла, в котором находится программа. Далее пять раз нажать Enter, написать *n* и нажать Enter. После этого можно посмотреть в режиме DOS выходную статистику в файле 2.txt.

2.1.6. Задания для самостоятельной работы

Задание 1

На ЭВМ поступает поток заявок, подчиненный равномерному закону. Обработка также подчинена равномерному закону. Описать процесс работы ЭВМ программой на GPSSR. Исследовать поведение модели заданной системы массового обслуживания (СМО) при вариации ее параметров.

Задание 2

Создать программу, моделирующую процесс прохождения заявок через прибор PR. Поступление заявок подчиняется равномерному закону, обработка – экспоненциальному закону распределения (функция распределения задана в виде табл. 2.3). Если у прибора нет возможности принять заявку, она становится в очередь OR. Исследовать поведение модели заданной системы массового обслуживания (СМО) при вариации ее параметров.

Таблица 2.3

Функция экспоненциального закона распределения

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.75	0.8	0.84	0.88
y	0	0.104	0.222	0.335	0.509	0.69	0.915	1.2	1.38	1.6	1.83	2.12
x	0.9	0.92	0.94	0.95	0.96	0.97	0.98	0.99	0.995	0.998	0.999	1
y	2.3	2.52	2.81	2.99	3.2	3.5	3.9	4.6	5.3	6.2	7	8

Задание 3

Рассматривается система с потерями. Число мест в очереди ограничено Z. В случае, если все Z мест в очереди заняты, заявка теряется. Поступление заявок подчинено экспоненциальному закону (табл. 1.3), обработка – равномерному закону.

2.2. Моделирование автоматизированных информационных систем с помощью GPSS

2.2.1. Сводное описание блоков языка GPSS

2.2.1.1. Вычислительные средства

FUNCTION – блок определения функции.

имя FUNCTION A,B

имя – имя функции;

A – аргумент функции для получения значений функции;

B – количество пар координат, определяющих функцию, задается таким образом:

СК – для непрерывных функций;

DK – для дискретных функций;

K – количество точек функции.

VARIABLE – блок определения целой переменной.

имя VARIABLE A

имя – имя переменной;

A – арифметическое выражение, определяющее переменную.

FVARIABLE – блок определения действительной переменной.

имя FVARIABLE A

имя – имя переменной;

A – логическое выражение.

BVARIABLE – блок определения булевой переменной.

имя BVARIABLE A

имя – имя переменной;

A – логическое выражение.

SAVEVALUE – блок изменения сохраняемых величин.

SAVEVALUE A,B,C

A – имя изменяемой сохраняемой величины. Если после A стоит знак <+>, то значение A увеличивается на B; если указан знак <->, то A уменьшается на B; если знак не указан, то A присваивается значение B;

B – параметр, используемый для модификации сохраняемой величины;

C – тип сохраняемой величины.

MATRIX – блок описания матриц.

имя MATRIX A,B,C

имя – имя матрицы;

A – тип матрицы (X – полнословная, H – полусловная);

B – количество строк в матрице;

C – количество столбцов в матрице.

MSAVEVALUE – блок изменения значения элемента матрицы.

MSAVEVALUE имя, A,B,C,D

имя – имя матрицы. Если после имени стоит знак <+>, то значение элемента матрицы увеличивается на C; если знак <->, то значение элемента матрицы уменьшается на C; если знака нет, то элементу матрицы приписывается значение C;

A – номер строки матрицы;

B – номер столбца матрицы;

C – величина, используемая для изменения значения элемента матрицы;

D – тип матрицы.

LOGIC – блок изменения логических переключателей.

LOGIC A B

A – оператор действия, который принимает значения:

R – сбросить логический переключатель;

S – установить логический переключатель;

I – инвертировать логический переключатель;

B – имя логического переключателя.

INITAL – блок установки начальных значений.

Установка значений сохраняемых величин:

INITAL A_i,B_i

A_i – имя сохраняемой величины;

B_i – начальное значение (может быть положительным и отрицательным).

Установка значений матриц:

INITAL A_i(C_i,D_i),B_i

A_i – имя матрицы;

C_i – номер строки матрицы;

D_i – номер столбца матрицы;

B_i – начальное значение.

2.2.1.2. Генерация и уничтожение транзактов

GENERATE – блок генерации транзактов.

GENERATE A,B,C,D,E,F,G

A – среднее значение интервала времени между последовательными прохождениями транзактов;

B – разброс интервала времени относительно A;

C – момент времени, в который должен появиться первый транзакт;

D – количество транзактов, которое должно быть сгенерировано, после чего генерация транзактов прекращается;

E – значение приоритета генерируемых транзактов, которое может лежать в диапазоне 0 – 127, причем большее значение соответствует более высокому приоритету;

F – количество параметров транзакта (≤100);

G – размер памяти, отводимый под один параметр транзакта, равен 4 байтам.

TERMINATE – блок уничтожения транзактов;

TERMINATE A

A – при уничтожении транзактов вычисляется СЧ = СЧ – N, где СЧ – счетчик завершений (задается в блоке START). При СЧ = 0 моделирование прекращается.

2.2.1.3. Изменение параметров транзактов

ASSIGN – блок изменения значений параметров.

ASSIGN A,B,C

A – номер параметра транзакта, подлежащего изменению. Если задан знак <+>, то прибавляется целая часть значения (BC), если задан <->, то вычитается; если знак не указан, то присваивается значение B;

B – величина, используемая для изменения значения параметра транзакта;

C – имя функции, используемой для модификации величины.

PRIORITY – блок и изменения приоритета.

PRIORITY A,B

A – значение приоритета, присваемое транзакту;

B – при наличии данного операнда интерпретатор переставляет транзакт в цепи текущих событий так, что он оказывается в конце нового приоритетного класса, и снова начинает просмотр цепи текущих событий.

2.2.1.4. Ансамбли транзактов

SPLIT – блок расщепления транзактов.

SPLIT A,B,C,D

A – число дополнительно порождаемых транзактов-«потомков»;

B – имя блока, куда будут направлены транзакты-«потомки».

C – номер параметра транзакта-«родителя».

D – число параметров, которое должен иметь каждый потомок.

ASSEMBLE – блок соединения транзактов.

ASSEMBLE A

A – количество членов ансамбля, объединяемых в один транзакт. Когда количество вошедших членов ансамбля станет равно A, транзакт, прибывший первым, выходит из блока ASSEMBLE.

GATHER – блок сбора транзакта.

GATHER A

A – количество членов ансамбля, накапливаемых в данном блоке. Когда количество вошедших членов ансамбля станет равно A, они выходят из блока GATHER.

MATCH – блок синхронизации транзактов.

MATCH A

A – имя блока, сопряженного с данным. Если сопряженный блок содержит транзакт, являющийся членом ансамбля, к которому принадлежит транзакт, вошедший в блок MATCH, то оба транзакта продолжают движение. В противном случае транзакт задерживается в блоке MATCH.

ADVANCE – блок задержки транзактов.

ADVANCE A,B

A – среднее значение интервала времени, на которое задерживается транзакт;

B – разброс интервала времени относительно A; вычисляется, как и в блоке GENERATE.

2.2.1.5. Приборы

SEIZE – блок занятия прибора.

SEIZE A

A – имя прибора, подлежащего занятию транзакта.

RELEASE – блок освобождения прибора.

RELEASE A

A – имя освобождаемого прибора.

PREEMT – блок захвата приборов.

PREEMT A,B,C,D,E

A – имя захватываемого прибора;

B – условие захвата прибора. Если указан параметр PR, то захват прибора происходит при условии, что вновь поступающий транзакт имеет более высокий приоритет.

C – имя блока, в который будет послан прерванный транзакт;

D – номер параметра прерванного транзакта, в который помещается значение времени, оставшееся транзакту до окончания обслуживания на приборе;

E – если задан параметр RE, то прерванный транзакт теряет право на автоматическое восстановление обработки в приборе.

RETURN – блок возврата захваченного прибора.

RETURN A

A – имя освобождаемого прибора, т.е. возврат прибора ранее прерванному транзакту.

2.2.1.6. Многоканальные устройства

STORAGE – блок описания ёмкости устройства.

имя STORAGE A

имя – имя многоканального устройства;

A – ёмкость многоканального устройства.

ENTER – блок входа в устройство.

ENTER A,B

A – имя многоканального устройства;

B – занимаемая ёмкость устройства.

LEAVE – блок выхода из устройства.

LEAVE A,B

A – имя многоканального устройства;

B – освобождаемая ёмкость устройства.

2.2.1.7. Блоки проверки условий

TRANSFER – блок передачи транзактов.

Безусловный режим:

TRANSFER A

A – имя блока, в который переходит транзакт.

Статический режим:

TRANSFER A,B,C

A – вероятность передачи транзакта на C;

B – имя блока, куда передаются транзакты с вероятностью (1 – A);

C – имя блока, куда передаются транзакты с вероятностью A.

Условный режим:

TRANSFER A,B,C

A – задает режим, при котором транзакт сначала пытается войти в блок B. Если вход невозможен, то транзакт пытается войти в блок C. Если транзакт не может войти и в этот блок, то транзакт остается в блоке TRANSFER;

B – имя блока;

C – имя блока.

SELECT – блок выбора элементов.

Режим отношения:

SELECT A B,MIN,MAX,C,D,E

B – номер параметра транзакта, в который записывается номер члена группы, соответствующий заданному условию;

MIN, MAX – наименьший и наибольший номера из множества членов просматриваемой группы;

C – значение, с которым должно сравниваться значение;

D – имя просматриваемого множества элементов;

E – имя блока, в который передается транзакт, если ни один элемент просматриваемого множества не отвечает заданному условию.

Режим работы минимального или максимального элемента:

SELECT A B,MIN,MAX,C

B, MIN, MAX, C – имеют тот же смысл, что и в режиме отношения;

A – если MIN, то ищется элемент с минимальным значением C, если MAX, то с максимальным значением C.

Логический режим:

SELECT A B, MIN, MAX, E

B, MIN, MAX, E – имеют тот же смысл, что и в режиме отношения;

A – логический указатель, задающий условие, которое должно выполняться, может принимать следующие значения:

LS – логический переключатель установлен;

LR – логический переключатель сброшен;

U – прибор используется;

NU – прибор не используется;

SF – многоканальное устройство заполнено;

SNF – многоканальное устройство не заполнено;

SE – многоканальное устройство пусто;

SNE – многоканальное устройство не пусто;

I – на приборе произошло прерывание;

NI – на приборе не произошло прерывание.

COUNT – блок подсчета элементов.

COUNT A B,MIN,MAX,C,D

B – номер параметра транзакта, в который заносится количество элементов, удовлетворяющих данному условию.

TEST – блок сравнения атрибутов.

TEST A B,C,D

B – имя первого стандартного атрибута;

C – имя второго стандартного атрибута;

D – имя блока, в который передается транзакт, если условие сравнения не выполняется;

A – оператор основания задает операцию сравнения:

G – B > C; NE – B <> C; GE – B ≥ C; LE – B ≤ C; E – B = C; L – B < C.

GATE – блок проверки состояния элементов.

Проверка состояния логических переключений:

GATE A B,C

B – имя логического переключателя;

C – имя блока, в который переходит транзакт, если проверяемое условие не выполняется;

A – логический указатель задает условие проверки:

LS – логический указатель установлен;

LR – логический указатель сброшен.

LOOP – блок организации цикла.

LOOP A,B

A – номер параметров транзакта, значение которого используется для организации количества повторений (параметр цикла). При входе транзакта в данный блок A уменьшается на 1. Если A становится равным 0, то транзакт переходит в следующий блок, иначе транзакт переходит в блок с именем B;

B – имя блока, в который переходит транзакт, если A<>0.

QUEUE – блок занятия очереди.

QUEUE A,B

A – имя очереди;

B – количество мест в очереди, занимаемое транзактом.

DEPART – блок освобождения очереди.

DEPART A,B

A – имя очереди;

B – количество мест в очереди, освобождаемое транзактом.

2.2.1.8. Построение гистограмм

TABLE – блок описания таблицы.

имя TABLE A,B,C,D,E

имя – имя таблицы;

A – имя переменной, значение которой табулируется. Если указан параметр IA, то осуществляется построение гистограмм интервалов времени между моментами поступления транзактов в данную точку. Если указан параметр RT, то осуществляется построение гистограмм интенсивности прихода транзакта в данную точку, причем интенсивность определяется относительно временного интервала.

B – левая граница первого интервала таблицы;

C – ширина интервалов таблицы;

D – количество интервалов таблицы, увеличенное на 2;

E – временной интервал для параметра RT.

QTABLE – блок описания таблицы времени пребывания в очереди.

имя QTABLE A,B,C,D

имя – имя таблицы;

A – имя очереди, для которой строится распределение времени пребывания транзактов в очереди.

MARK – блок отметки

MARK A

A – номер параметра транзакта, в который заносится момент времени входа транзакта в данный блок.

TABULATE – блок табулирования

TABULATE A,B

A – имя таблицы, в которую заносится табулируемая величина, указанная в блоке TABLE, в момент входа транзакта в данный блок;

B – весовой коэффициент, задающий число раз, которое табулируемая величина должна занести в таблицу при каждом входе в данный блок. При использовании данного параметра в блоке TABLE параметр K должен быть задан как WK.

2.2.1.9. Цепи пользователя

LINK – блок ввода транзакта в цепь пользователя.

LINK A,B,C

A – имя цепи пользователя;

B – критерий присоединения транзакта к цепи пользователя. B может принимать следующие значения:

FITO – встать в конец цепи;

LITO – встать в начало цепи;

P_i – войти в цепь непосредственно перед транзактом с большим значением *i*-го параметра;

C – имя блока, куда переходит транзакт, если он присоединяется к цепи пользователя.

UNLINK – блок вывода транзакта из цепи пользователя

UNLINK A,B,C,D,E,F

A – имя цепи пользователя;

B – имя блока, в который переходят выведенные из цепи транзакты;

C – число вводимых транзактов. Если задан параметр ALL, то выводятся все транзакты, отвечающие условиям:

D, E – определяют условия вывода транзактов из цепи пользователя;

F – имя блока, куда переходит транзакт – инициатор вывода, если из цепи не выводится ни один транзакт.

2.2.1.10. Служебные карты

SIMULATE – блок моделирования.

Этот блок должен быть первым блоком программы модели. Если он отсутствует, то выполнение модели не производится.

END – блок конца программы.

Этот блок должен быть последним.

START – блок начала моделирования.

START A,B,C,D

A – начальное значение счетчика числа завершений;

B – признак подавления печати. Если задан параметр NP, то стандартная печать в конце моделирования производиться не будет;

C – задает число завершений, через которое будет выдаваться промежуточная печать;

D – признак печати цепей. Печать цепей производится, если на месте D стоит 1.

2.2.2. Примеры распечатки программ

Заголовок имеет вид :

block number *loc operation A,B,C,D,E,F,G Comments card number

Поля заголовка имеют следующие значения:

block number – номер блока. Все блоки в программе должны быть пронумерованы. В этом поле указываются порядковые номера блоков (но не операторов);

*loc (location) – поле имени. Здесь указываются имена блоков и таблиц.

Символ * говорит о том, что данная строка программы является комментарием или пустой строкой;

operation A,B,C,D,E,F,G – поле операций и поле операндов (помечено символами A, B и т.д.). В этом поле указываются блоки, операторы и их операнды;

comments – комментарии. В нем записываются необходимые пояснения к программе;

card number – это поле содержит порядковые номера всех строк программы.

Основные результаты, получаемые при моделировании, включают в себя модель после ассемблирования и список блоков.

Ассемблирование – это процесс трансляции с исходного языка в некоторый промежуточный объектный код. После ассемблирования все символические имена заменяются на числовые и модель принимает, например, следующий вид:

```
*
  1  GENERATE      18,6
  2  QUEUE         1
  3  SEIZE         1
  4  DEPART        1
  5  ADVANCE       16,4
  6  RELEASE       1
  7  TERMINATE
*
  8  GENERATE      480
  9  TERMINATE     1
*
     START         100
```

Более подробно данная модель рассматривается ниже.

BLOCK COUNTS – СПИСОК БЛОКОВ – может иметь следующий вид:

Block	current	total
1	0	27

2	1	27
3	0	26
4	0	26
5	1	26
6	0	25
7	0	25
8	0	1
9	0	1

Block (блок) – номер блока;

Current (текущее) – счетчик текущего содержимого, т.е. это есть счетчик транзактов, находящихся в соответствующих блоках в момент завершения моделирования;

Total (общее) – общее число входов. Счетчик входов является счетчиком общего числа транзактов, которые вошли в соответствующие блоки в течение времени моделирования, включая также те из них, которые все еще находятся в блоках (если такие имеются). Вообще, каждой задаче соответствует определенный набор выходных параметров и эти наборы могут быть различными в зависимости от поставленной задачи.

Рассмотрим простейший пример – моделирование системы обслуживания с одним прибором и очередью.

Задача. Интервалы прихода клиентов в парикмахерскую с одним креслом распределены равномерно в интервале 18 ± 6 мин. Время стрижки также распределено равномерно: 16 ± 4 мин. Клиенты приходят, стригутся в порядке «первым пришел – первым обслужен» и затем уходят.

Модель парикмахерской на GPSS должна обеспечить сбор статистических данных об очереди. Необходимо промоделировать работу в течение 8 часов.

Распечатка программы:

block num-ber	*loc	Operation	A,B,C,D,E,F,G	Comments	card number
		SIMULATE			1
	*				2
1		GENERATE	18,6	приход клиентов	3
2		QUEUE	joeq	присоединение к очереди	4
3		SEIZE	joe	переход в кресло парикмахера	5
4		DEPART	joeq	уход из очереди	6

```
5          ADVANCE    16,4     работа парикмахера                  7
6          RELEASE    joe      уход из кресла парикмахера          8
7          TERMINATE           уход из парикмахерской              9
       *                                                           10
8          GENERATE   480      моделировать 480 единиц времени     11
9          TERMINATE  1        завершение прогона                  12
       *                                                           13
           START      1                                            14
           END                                                     15
```

Пример *выходных данных*:

relative clock 480 absolute clock 480

Facility	average untili-zation	number entries	avarage time/ tran	seizing trans. no.	preempti ng trans. no.		
Joe	0.860	26	15.884	3			
queue	maxi-mum contents	avarage cont-ents	total entries	zero entries	percent zeros	avarage time/ trans	$avara-ge time/ trans
joeq	1	0.160	27	12	44.4	2.852	5.133

По завершении моделирования интерпретатор GPSS автоматически распечатывает некоторое количество информации о поведении модели. Эта информация включает статистические данные по каждому из элементов, используемых в модели, т.е. по каждому из приборов, очередей и т.д. Для того чтобы иметь представление о работе модели на основании результатов моделирования, ниже для данного примера приводится описание всех выходных параметров и русский эквивалент их названий.

Описание значений *выходных параметров*

1. RELATIVE CLOCK 480 и ABSOLUTE CLOCK 480

ОТНОСИТЕЛЬНОЕ ВРЕМЯ и АБСОЛЮТНОЕ ВРЕМЯ – в этом и других примерах данные параметры означают, что моделирование завершилось в момент модельного времени, равного 480.

2. FACILITY (полное обозначение – facility simbols corresponding numbers).

Joe ПРИБОР (словарь символов для приборов) – здесь указывается перечень приборов. В перечне представлены все символические имена приборов (или присвоенные им числовые эквиваленты), используемых в модели.

3. QUEUE (queue simbols corresponding numbers).

Joeq ОЧЕРЕДЬ (словарь символов для очередей) – полностью аналогично FACILITY.

Выходные данные для прибора joe

4. AVARAGE UTILIZATION 0.860

СРЕДНЯЯ НАГРУЗКА – время, в течение которого прибор был занят, т.е. в течение 86 % модельного времени.

5. NUMBER ENTRIES 26

ЧИСЛО ВХОДОВ – число обслуживаний, т.е. прибор был занят обслуживанием 26 раз.

6. AVARAGE TIME/TRAN 15.884

СРЕДНЕЕ ВРЕМЯ/ТРАНЗ – среднее время интервала обслуживания, т.е. средняя продолжительность обслуживания прибора joe равна 15,884 мин.

7. SEIZING TRANS.NO 3

НОМЕР ОБСЛУЖИВАЕМОГО ТРАНЗАКТА – транзакт номер 3 находился на обслуживании прибором joe в момент завершения моделирования.

8. PREEMPTING TRANS.NO

НОМЕР ЗАХВАТИВШЕГО ТРАНЗАКТА – транзакта, который захватил прибор, при завершении моделирования нет.

Выходные данные для очереди joeq

9. MAXIMUM CONTENTS 1

МАКСИМАЛЬНОЕ СОДЕРЖИМОЕ – наибольшее значение содержимого очереди, зарегистрированное в течение моделирования, т.е. в очереди никогда не было более одного клиента.

10. AVARAGE CONTENTS 0.160

СРЕДНЕЕ СОДЕРЖИМОЕ – среднее значение содержимого очереди, т.е. среднее число клиентов, находившихся в очереди, равно 0,16.

11. TOTAL ENTRIES 27

ОБЩЕЕ ЧИСЛО ВХОДОВ – общее число элементов содержимого, вошедших в очередь, т.е. всего в очередь вставало 27 клиентов.

12. ZERO ENTRIES 12

НУЛЕВЫЕ ВХОДЫ – общее число входов в очередь без последующего ожидания, т.е. среди 27 клиентов 12 сразу стали обслуживаться.

13. PERCENT ZEROS 44.4

ПРОЦЕНТ НУЛЕВЫХ – из общего числа входов в очередь 44,4 % было нулевых.

14. AVARAGE TIME/TRAN 2.851

СРЕДНЕЕ ВРЕМЯ/ТРАНЗ – среднее время, проведенное в очереди, с учетом нулевых входов, т.е. среднее время нахождения клиента в очереди равно 2,851 мин.

15. $AVARAGE TIME/TRANS 5.133

$СРЕДНЕЕ ВРЕМЯ/ТРАНЗ – среднее время нахождения в очереди на один ненулевой вход, т.е. каждый клиент, который был в очереди, ждал 5,133 мин.

16. TABLE NUMBER

ИМЯ ТАБЛИЦЫ – имя таблицы, в которой представлено распределение времени ожидания для этой очереди. В данном примере не используется.

17. CURRENT CONTENTS 1

ТЕКУЩЕЕ СОДЕРЖИМОЕ – текущее значение содержимого очереди, т.е. в момент завершения моделирования один транзакт находился в очереди.

2.2.3. Пример моделирования автоматизированной информационной системы

Сущность машинного моделирования системы состоит в проведении на вычислительной машине эксперимента с моделью, которая представляет собой некоторый программный комплекс, описывающий формально и (или) алгоритмически поведение элементов системы в процессе ее функционирования, т.е. в их взаимодействии друг с другом и внешней средой.

Основные этапы моделирования системы [2.4]:

- построение концептуальной модели системы и ее формализация;
- алгоритмизация модели системы и ее машинная реализация;
- получение и интерпретация результатов моделирования системы.

На этапе построения концептуальной модели и ее формализации проводится исследование моделируемого объекта с точки зрения выделения основных составляющих процесса его функционирования, определяются необходимые аппроксимации и получается обобщенная схема модели системы, которая преобразуется в машинную модель на втором этапе моделирования путем последовательной алгоритмизации и программирования модели. Последний, третий этап моделирования системы, сводится к проведению рабочих расчетов на ЭВМ согласно полученному плану, получению и интерпретации результатов моделирования системы с учетом воздействия внешней среды.

Задание

В студенческом машинном зале расположены две ЭВМ и одно устройство подготовки данных (УПД). Студенты приходят с интервалом в 8 ± 2 мин, и

треть из них хочет использовать УПД и ЭВМ, а остальные только ЭВМ. Допустимая очередь в машинный зал составляет четыре человека, включая работающего на УПД. Работа на УПД занимает 8 ± 1 мин, а на ЭВМ – 17 мин. Кроме того, 20 % работавших на ЭВМ возвращаются для повторного использования УПД и ЭВМ. Смоделировать работу машинного зала в течение 6 ч. Определить загрузку УПД, ЭВМ и вероятности отказа в обслуживании вследствие переполнения очереди. Определить соотношение желающих работать на ЭВМ и на УПД в очереди.

2.2.3.1. Построение концептуальной модели системы и ее формализация

2.2.3.1.1. Формулировка цели и постановка задачи машинного моделирования системы

Необходимо исследовать работу студенческого машинного зала. В качестве *цели моделирования* выберем изучение функционирования системы, а именно оценивание ее характеристик с точки зрения эффективности работы системы, т.е. будет ли она простаивать, работать на износ или работать с запасом. В качестве цели эффективного функционирования системы целесообразно выбрать максимизацию загрузки УПД и ЭВМ и одновременно минимизацию вероятности отказа в обслуживании вследствие переполнения очереди к УПД и ЭВМ.

С учетом имеющихся ресурсов в качестве *метода решения задачи* выберем метод имитационного моделирования, позволяющий не только анализировать характеристики модели, но и проводить структурный, алгоритмический и параметрический синтез модели на ЭВМ при заданных критериях оценки эффективности и ограничениях.

Постановка задачи исследования функционирования студенческого машинного зала как системы, состоящей из УПД и двух ЭВМ и содержащей обратную связь, представлена в задании, из которого следует, что необходимо определить:

- загрузку УПД и каждой ЭВМ;
- загрузку очереди к УПД и каждой ЭВМ;
- вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ;
- соотношение желающих работать на ЭВМ и на УПД в очереди.

Пересмотр начальной постановки задачи исследования не предусмотрен.

2.2.3.1.2. Анализ задачи моделирования системы

В качестве *критерия оценки эффективности* процесса функционирования системы целесообразно выбрать вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ, которая должна быть минимальной, при этом загрузка УПД и каждой ЭВМ должна быть максимальной.

Соотношение загрузки каждой ЭВМ и УПД должно быть в среднем одинаковым, чтобы каждое устройство было задействовано равноценно. В качестве еще одного традиционного критерия оценки эффективности процесса функционирования системы можно выбрать минимальное время обслуживания заявок в системе в целом при максимальном количестве обслуженных заявок.

Экзогенные (независимые) переменные модели:

- интервал времени (интенсивность) прихода студентов в зал;
- допустимая очередь в машинный зал;
- время работы студентов на ЭВМ, УПД.

Эндогенные (зависимые) переменные модели:

- загрузка УПД и каждой ЭВМ;
- загрузка очереди к УПД и каждой ЭВМ;
- количество студентов, работающих на каждой ЭВМ и УПД;
- количество студентов, желающих повторно работать на УПД и ЭВМ;
- количество студентов, которые получили отказ в обслуживании вследствие переполнения очереди к УПД или ЭВМ;
- вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ;
- соотношение желающих работать на ЭВМ и на УПД в очереди.

При построении математической имитационной модели процессов функционирования системы будем использовать непрерывно-стохастический подход на примере типовой *Q*-схемы, потому что исследуемая система – студенческий машинный зал – может быть представлена как система массового обслуживания с непрерывным временем обработки параметров при наличии случайных факторов.

Формализовав процесс функционирования исследуемой системы в абстракциях *Q*-схемы, на втором этапе алгоритмизации модели и ее машинной реализации выберем язык имитационного моделирования, потому что высокий уровень проблемной ориентации языка значительно упростит программирование, а специально предусмотренные в нем возможности сбора, обработки и вывода результатов моделирования позволят быстро и подробно проанализировать возможные исходы имитационного эксперимента с моделью. Для получения полной информации о характеристиках процесса функционирования системы необходимо будет провести полный факторный эксперимент, который позволит определить, насколько эффективно функционирует система, и выдать рекомендации по ее усовершенствованию.

2.2.3.1.3. Определение требований к исходной информации об объекте моделирования и организация ее сбора

Вся необходимая информация о системе и внешней среде представлена в задании и не требует предварительной обработки.

2.2.3.1.4. Выдвижение гипотез и принятие предположений

Для заполнения пробелов в понимании задачи исследования, а также проверки возможных результатов моделирования при проведении машинного эксперимента выдвигаем следующие *гипотезы*:

- загрузка УПД будет меньше загрузки ЭВМ, т.к. интервал времени работы студентов на УПД меньше, чем на ЭВМ;
- вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ будет больше 0,6, т.к., несмотря на то, что время работы обеих ЭВМ (17 мин) соответствует интенсивности прихода студентов в зал (8 ± 2 мин), которые равномерно рассредоточиваются между двумя ЭВМ, часть студентов (30 %) дополнительно желает работать на УПД. При этом время работы на УПД (8 ± 1 мин) соответствует интенсивности поступления студентов и, кроме того, 20 % работавших на ЭВМ возвращаются для повторного использования УПД и ЭВМ;
- соотношение желающих работать на ЭВМ и на УПД в очереди составит 2:1, поскольку после работы на УПД студент обязательно будет работать на одной из ЭВМ, но, несмотря на то, что время работы на УПД меньше, часть студентов желает повторно поработать именно на УПД.

Для упрощения модели можно выдвинуть следующие *предположения*:

- время перехода студента с УПД на ЭВМ равно нулю;
- студент, желающий повторно работать на УПД и ЭВМ, считается уже находящимся в зале и место в очереди в зал он не занимает;
- если студент не желает работать на УПД, то он занимает любую свободную ЭВМ;
- после работы на УПД студент занимает только вторую ЭВМ;
- 20 % студентов возвращаются для повторного использования УПД и ЭВМ независимо от того, пользовались ли они ранее УПД.

2.2.3.1.5. Определение параметров и переменных модели

Входные переменные модели:

- интервал времени (интенсивность) прихода студентов в зал, $T_{ПР} \pm \Delta T_{ПР}$, где $T_{ПР}$ – средний интервал времени между приходом студентов в машинный зал, $\Delta T_{ПР}$ – половина интервала, в котором равномерно распределено значение, единица измерения – минута.

Если интенсивность прихода студентов в зал будет меньше времени работы студентов на УПД и ЭВМ, то загрузка системы в целом будет возрастать, и, как следствие, будет увеличиваться количество студентов, которые получат отказ в обслуживании.

Выходные переменные модели:

- количество студентов, отработавших на ЭВМ или УПД и ЭВМ за заданный интервал времени работы машинного зала, $N_{ОБС}$, единица измерения – количество студентов;
- количество студентов, которые получили отказ в обслуживании вследствие переполнения очереди к УПД или ЭВМ за заданный интервал времени работы машинного зала, $N_{ОТК}$, единица измерения – количество студентов.

Параметры модели:

- допустимая очередь в машинный зал, $L_{ЗАЛ}$, единица измерения – количество студентов;
- время работы студентов на первой и второй ЭВМ, $T_{ЭВМ1}$, $T_{ЭВМ2}$, единица измерения – минута;
- время работы студентов на УПД, $T_{УПД} \pm \Delta T_{УПД}$, где $T_{УПД}$ – среднее время работы студентов на УПД, $\Delta T_{УПД}$ – половина времени, в котором равномерно распределено значение, единица измерения – минута;
- среднее время обслуживания студентов в машинном зале, $T_{ОБС}$, единица измерения – минута;
- загрузка УПД, $Z_{УПД}$, единица измерения – относительная единица;
- загрузка первой и второй ЭВМ, $Z_{ЭВМ1}$, $Z_{ЭВМ2}$, единица измерения – относительная единица;
- загрузка очереди к УПД, $Z_{ОЧ.УПД}$, единица измерения – относительная единица;
- загрузка очереди к первой и второй ЭВМ, $Z_{ОЧ.ЭВМ1}$, $Z_{ОЧ.ЭВМ2}$, единица измерения – относительная единица;
- количество студентов, желающих работать только на ЭВМ, $N_{ЭВМ}$, единица измерения – количество студентов;
- количество студентов, желающих работать не только на ЭВМ, но и на УПД, $N_{УПД}$, единица измерения – количество студентов;
- количество студентов, желающих повторно работать на УПД и ЭВМ, $N_{ПР}$, единица измерения – количество студентов;
- вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ, $P_{ОТК}$, единица измерения – относительная единица.

Уменьшение допустимой очереди в машинный зал и (или) увеличение времени работы студентов на УПД и ЭВМ, и (или) увеличение количества студентов, желающих повторно работать на УПД и ЭВМ, будет приводить к увеличению загрузки системы в целом и, как следствие, к увеличению количества студентов, которые получат отказ в обслуживании.

Воздействия внешней среды отсутствуют.

2.2.3.1.6. Установление основного содержания модели

На основе анализа исходных данных и выдвинутых гипотез можно сделать вывод о том, что процессы, происходящие в моделируемой системе, являются процессами массового обслуживания, поэтому эти процессы целесообразно описать на языке *Q*-схем.

2.2.3.1.7. Обоснование критериев оценки эффективности системы

Для оценки качества процесса функционирования моделируемой системы сформируем на основании анализа задачи моделирования системы *функцию поверхности отклика* в исследуемой области изменения параметров и переменных как совокупность критериев оценки эффективности. Эта функция позволит определить экстремумы реакции системы.

$$\begin{cases} Z_{\text{ЭВМ1}} = f(T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{ЭВМ1}}, T_{\text{ЭВМ2}}, N_{\text{ПР}}), \\ Z_{\text{ЭВМ2}} = f(T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{ЭВМ1}}, T_{\text{ЭВМ2}}, N_{\text{ПР}}), \\ Z_{\text{УПД}} = f(T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{УПД}} \pm \Delta T_{\text{УПД}}, N_{\text{ПР}}), \\ N_{\text{ОТК}} = f(Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}), \\ P_{\text{ОТК}} = f(Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}), \\ T_{\text{ОБС}} = f(T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}), \\ N_{\text{ОБС}} = f(T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}). \end{cases}$$

2.2.3.1.8. Определение процедур аппроксимации

Для аппроксимации реальных процессов, протекающих в системе, воспользуемся процедурой определения средних значений выходных переменных, поскольку в системе имеются случайные значения переменных и параметров.

2.2.3.1.9. Описание концептуальной модели системы

Концептуальная модель исследуемой системы представлена в виде структурной схемы (см. рис. 2.5), состоящей из одного входного потока x – студенты, приходящие в машинный зал; трех выходных потоков: y_1, y_2 – студенты, отработавшие в машинном зале на соответствующей ЭВМ, и y_3 – студенты, которым не хватило места в зале; трех блоков – устройств (УПД, ЭВМ1, ЭВМ2), связанных между собой согласно условию задачи.

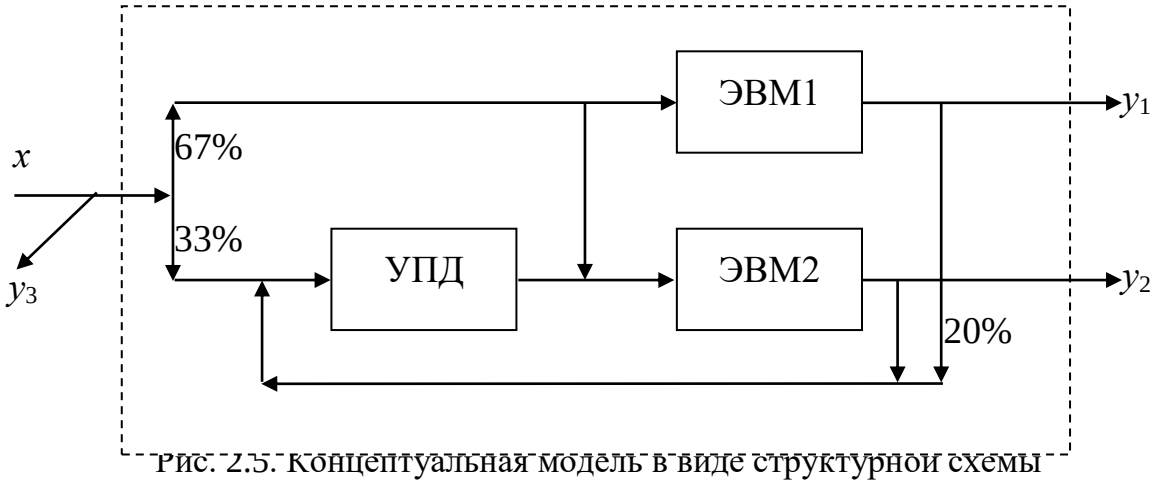


Рис. 2.5. Концептуальная модель в виде структурной схемы

Целевая функция модели системы:

$$\begin{cases} P_{\text{ОТК}}(Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}) \to \min\limits_{T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{ЭВМ1}}, T_{\text{ЭВМ2}}, T_{\text{УПД}} \pm \Delta T_{\text{УПД}}, N_{\text{ПР}}}, \\ Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}} \to \max\limits_{T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{ЭВМ1}}, T_{\text{ЭВМ2}}, T_{\text{УПД}} \pm \Delta T_{\text{УПД}}, N_{\text{ПР}}} \bigcup \\ \bigcup Z_{\text{ОЧ.ЭВМ1}}, Z_{\text{ОЧ.ЭВМ2}}, Z_{\text{ОЧ.УПД}} \to \min\limits_{T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, T_{\text{ЭВМ1}}, T_{\text{ЭВМ2}}, T_{\text{УПД}} \pm \Delta T_{\text{УПД}}, N_{\text{ПР}}}, \\ Z_{\text{ЭВМ1}} \cong Z_{\text{ЭВМ2}} \cong Z_{\text{УПД}}, \\ T_{\text{ОБС}} \to \min\limits_{T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}} \bigcup N_{\text{ОБС}} \to \max\limits_{T_{\text{ПР}} \pm \Delta T_{\text{ПР}}, L_{\text{ЗАЛ}}, Z_{\text{ЭВМ1}}, Z_{\text{ЭВМ2}}, Z_{\text{УПД}}}. \end{cases}$$

В качестве типовой математической схемы применяется *Q*-схема, состоящая из одного источника (И), накопителя (Н), трех каналов ($К_1$, $К_2$, $К_3$), шести клапанов (см. рис. 2.6). Заявки (студенты, приходящие в машинный зал) в систему поступают от источника И с интервалом 8 ± 2 мин в накопитель Н с емкостью $L_{\text{Н}}$, равной 3, поскольку по условию очередь в машинный зал может быть только из 4 человек, включая заявку в канале $К_2$ (УПД). Канал $К_1$ соответствует ЭВМ1, канал $К_2$ – УПД, канал $К_3$ – ЭВМ2. От источника заявки поступают в клапан 1, который управляется накопителем Н. В случае отсутствия места в накопителе заявки получают отказ $N_{\text{ОТК}}$. От накопителя Н заявки поступают в клапан 2, который условно управляется источником, распределяющим заявки между каналом $К_1$ (60 %) и каналами последовательной обработки $К_2$ и $К_3$ (30 %). Обработка (задержка) заявки в канале $К_1$ занимает 17 мин. Клапан 3 управляется каналом $К_1$, в случае его занятия заявка посылается на канал $К_2$. Обработка (задержка) заявки в канале $К_2$ занимает 8 ± 1 мин. Клапан 4 принимает заявки от клапанов 2 и 6, управляется каналом $К_2$, в случае его занятия заявка встает в очередь. Обработка (задержка) заявки в канале $К_3$ занимает 17 мин. Клапан 5 принимает заявки от клапана 3 и канала $К_2$, управляется каналом $К_3$, в случае его занятия заявка встает в очередь. Клапан 6 принимает заявки от каналов $К_1$ и $К_3$, управляется соответствующим каналом, при этом 20 % заявок не уничтожается,

а поступает на клапан 4 для повторного обслуживания в каналах $К_2$ и $К_3$. Остальные 80 % заявок считаются обслуженными $N_{ОБС}$ и уничтожаются.

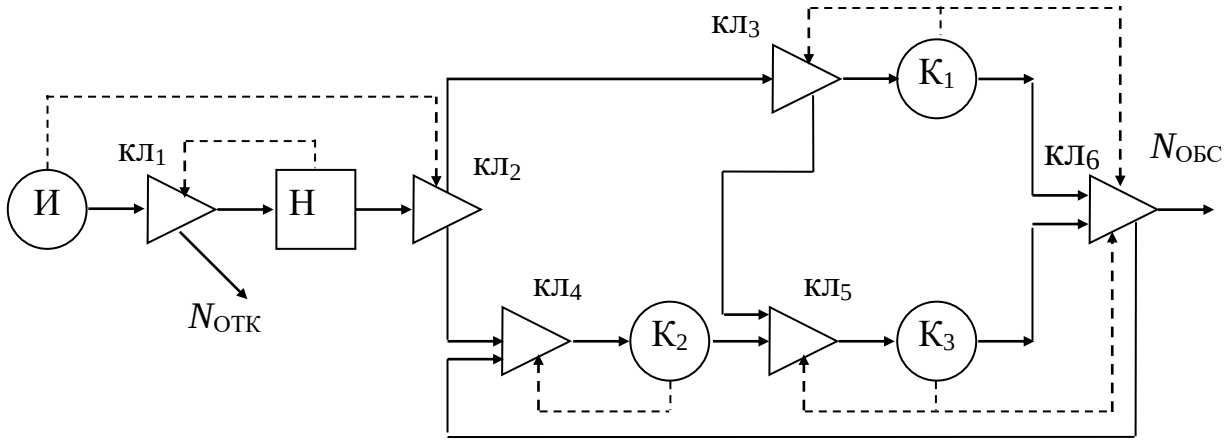


Рис. 2.6. Концептуальная модель в виде *Q*-схемы

Формальная модель системы:

Q = { И, Н, $К_1$, $К_2$, $К_3$, $N_{ОБС}$, $N_{ОТК}$, $кл_1$, $кл_2$, $кл_3$, $кл_4$, $кл_5$, $кл_6$, $L_Н$ = 3 }.

Согласно разработанной концептуальной модели окончательные гипотезы и предположения совпадают с ранее принятыми. Выбранная процедура аппроксимации определения средних значений выходных переменных соответствует реальным случайным процессам, протекающим в системе массового обслуживания.

2.2.3.1.10. Проверка достоверности концептуальной модели

Проверка достоверности концептуальной модели должна включать:

а) проверку замысла модели;

б) оценку достоверности исходной информации;

в) рассмотрение задачи моделирования;

г) анализ принятых аппроксимаций;

д) исследование гипотез и предположений.

Данный пункт предлагается проработать самостоятельно.

2.2.3.2. Алгоритмизация модели системы и ее машинная реализация

2.2.3.2.1. Построение логической схемы модели

Логическая схема модели представлена на рис. 2.7. После генерации заявок в источнике И (блок 1) осуществляется проверка на наличие мест в накопителе Н (блок 2).

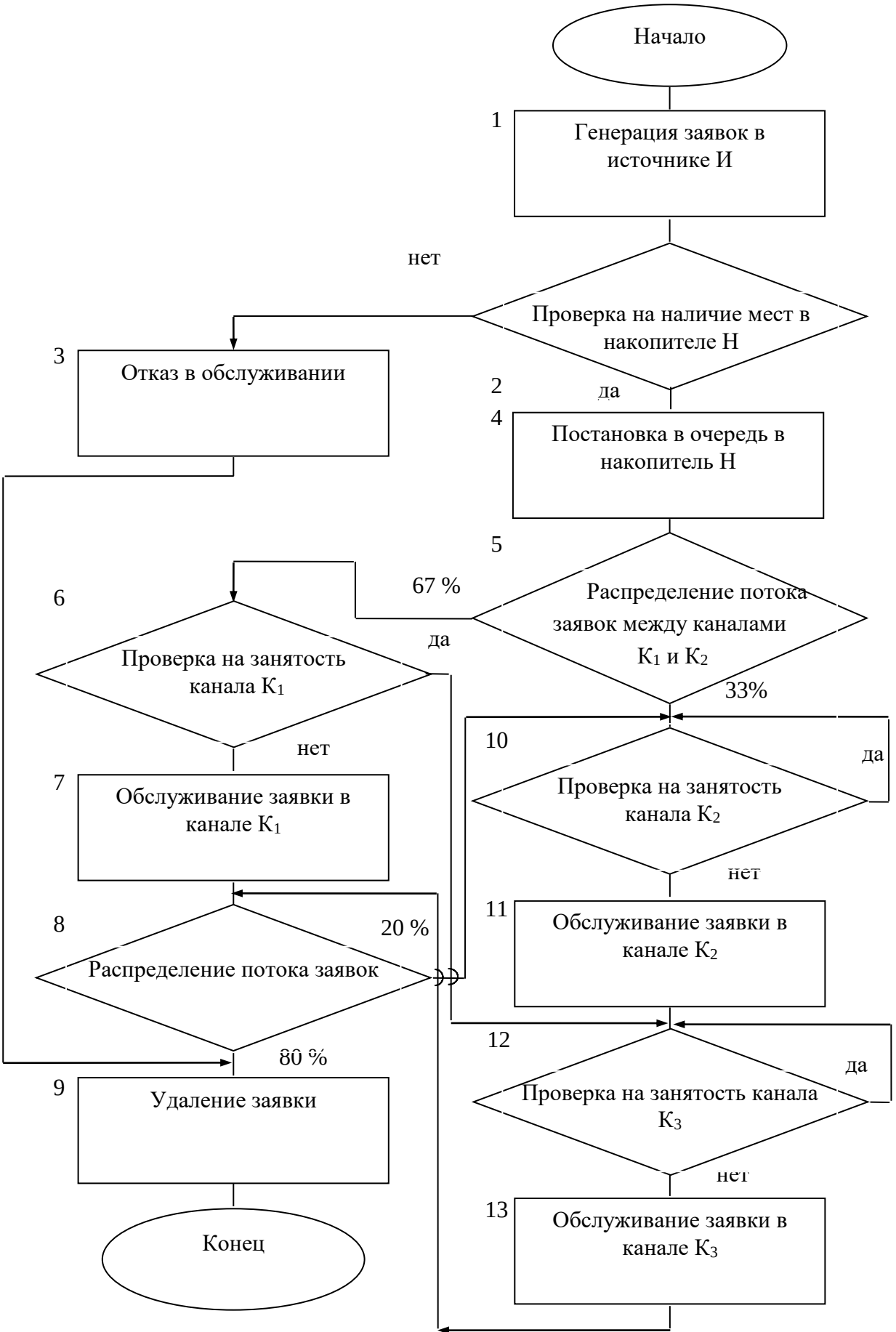


Рис. 2.7. Логическая схема

При отсутствии мест заявке будет отказано в обслуживании (блок 3), она будет удалена (блок 9) и покинет систему. Если место в накопителе Н есть, то заявка становится в очередь в накопителе (блок 4). Далее осуществляется равномерное распределение потока заявок между каналами $К_1$ и $К_2$ (блок 5): 60 % заявок поступает на проверку занятости канала $К_1$ (блок 6), 30 % заявок – на проверку занятости канала $К_2$ (блок 10). Если канал $К_1$ занят, то заявки поступают на проверку занятости канала $К_3$ (блок 12). Если канал $К_1$ свободен, то заявки обслуживаются в канале $К_1$ (блок 7). После этого осуществляется равномерное распределение потока заявок (блок 8): 80 % заявок поступает на удаление (блок 9) и покидает систему, 20 % заявок поступает на проверку занятости канала $К_2$ (блок 10).

В случае занятости канала $К_2$ продолжается проверка на занятость (блок 10). Если канал $К_2$ свободен, то заявки обслуживаются в канале $К_2$ (блок 11). Далее аналогично происходит проверка на занятость канала $К_3$ (блок 12), и если канал $К_3$ свободен, то заявки обслуживаются в канале $К_3$ (блок 13). Для того чтобы определить загруженность (или простои) каналов $К_2$ и $К_3$, можно проанализировать статистические данные, касающиеся очереди перед соответствующими каналами. После обслуживания заявки в канале $К_3$ она поступает в блок равномерного распределения потока заявок (блок 8): 80 % заявок поступает на удаление (блок 9) и покидает систему, 20 % заявок поступает на проверку занятости канала $К_2$ (блок 10).

2.2.3.2.2. Получение математических соотношений

Для построения машинной модели системы в комбинированном виде, т.е. с использованием аналитико-имитационного подхода, необходимо часть процессов в системе описать аналитически, а другую часть сымитировать соответствующими алгоритмами. На данном этапе построения аналитической модели зададим математические соотношения в виде явных функций.

Вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ ($P_{\text{ОТК}}$) зависит от количества студентов, отработавших на ЭВМ или УПД и ЭВМ ($N_{\text{ОБС}}$), и количества студентов, которые получили отказ в обслуживании вследствие переполнения очереди к УПД или ЭВМ ($N_{\text{ОТК}}$) за заданный интервал времени работы машинного зала:

$$P_{\text{ОТК}} = N_{\text{ОБС}} /(N_{\text{ОТК}} + N_{\text{ОБС}}).$$

Среднее время обслуживания студентов в машинном зале ($T_{\text{ОБС}}$) находится в пределах времени работы студентов на первой ЭВМ ($T_{\text{ЭВМ1}}$) и суммарного времени работы студентов на УПД ($T_{\text{УПД}}$), второй ЭВМ ($T_{\text{ЭВМ2}}$), повторного времени работы студентов на УПД и второй ЭВМ с вероятностью 20 %, а также времени ожидания в очереди к УПД и второй ЭВМ, поскольку студенты, желающие повторно работать, остаются в машинном зале независимо от их количества.

$$\begin{cases} \min T_{\text{ОБС}} = T_{\text{ЭВМ1}}, \\ \max T_{\text{ОБС}} = T_{\text{УПД}} + T_{\text{ЭВМ2}} + 0{,}2(T_{\text{УПД}} + T_{\text{ЭВМ2}}) + T_{\text{ОЧ}}. \end{cases}$$

Загрузку УПД, ЭВМ и соотношение желающих работать на ЭВМ и УПД в очереди в виде явных функций записать трудно. Эти величины определим с помощью языка имитационного моделирования.

2.2.3.2.3. Проверка достоверности модели системы

На данном подэтапе достоверность модели системы проверяется по следующим показателям:

а) возможности решения поставленной задачи;

б) точности отражения замысла в логической схеме;

в) полноте логической схемы модели;

г) правильности используемых математических соотношений.

Данный пункт предлагается проработать самостоятельно, ответив на вопрос, насколько логическая схема модели системы и используемые математические соотношения отражают замысел модели, сформированный на первом этапе.

2.2.3.2.4. Выбор инструментальных средств моделирования

В нашем случае для проведения моделирования системы массового обслуживания с непрерывным временем обработки параметров при наличии случайных факторов необходимо использовать ЭВМ с применением языка имитационного моделирования GPSS, т.к. в настоящее время самым доступным средством моделирования систем является ЭВМ, а применение простого и доступного языка имитационного моделирования GPSS (http://www.gpss.ru) позволяет получить информацию о функции состояний $z_i(t)$ системы, анализируя непрерывные процессы функционирования системы только в «особые» дискретные моменты времени при смене состояний системы благодаря моделирующему алгоритму, реализованному по «принципу особых состояний» (принцип δz). Кроме того, высокий уровень проблемной ориентации языка GPSS значительно упростит программирование, специально предусмотренные в нем возможности сбора, обработки и вывода результатов моделирования позволят быстро и подробно проанализировать возможные исходы имитационного эксперимента с моделью.

2.2.3.2.5. Составление плана выполнения работ по программированию

Выбранный язык имитационного моделирования GPSS имеет три версии: MICRO-GPSS Version 88-01-01, GPSS/PC Version 2, GPSS World Students Version 4.3.5. MICRO-GPSS имеет DOS-интерфейс, чувствителен к стилю написания программы (количеству пробелов между операндами, длине меток и имен и др.), не содержит текстового редактора. GPSS/PC лишен указанных недостатков, однако интерпретатор GPSS World Students имеет ряд преимуществ перед ним, например наличие интерфейса Windows, пошагового

отладчика, возможность сбора и сохранения в файлах различной статистической информации, визуальный ввод команд. Поэтому для разработки модели был выбран именно интерпретатор GPSS World Students.

Для моделирования достаточно использовать ЭВМ типа IBM/PC, применение специализированных устройств не требуется. В программное обеспечение ЭВМ, на которой проводится моделирование, должны входить операционная система Windows (версия 9X и выше) и интерпретатор GPSS. Затраты оперативной и внешней памяти незначительны, и необходимости в их расчете при современном уровне техники нет. Затраты времени на программирование и отладку программы на ЭВМ зависят только от уровня знаний языка и имеющихся навыков исследователя.

2.2.3.2.6. Спецификация и построение схемы программы

К программе на языке имитационного моделирования GPSS согласно спецификации программы предъявляются традиционные требования: структурированность, читабельность, корректность, эффективность и работоспособность, которые предлагается проработать самостоятельно.

Спецификация постановки задачи – определить количество студентов, отработавших на ЭВМ или УПД и ЭВМ за заданный интервал времени работы машинного зала ($N_{ОБС}$), и количество студентов, которые получили отказ в обслуживании вследствие переполнения очереди к УПД или ЭВМ за заданный интервал времени работы машинного зала ($N_{ОТК}$). В качестве исходных данных задаются интервал времени (интенсивность) прихода студентов в машинный зал ($T_{ПР} \pm \Delta T_{ПР}$), допустимая очередь в машинный зал ($L_{ЗАЛ}$), время работы студентов на первой и второй ЭВМ ($T_{ЭВМ1}$, $T_{ЭВМ2}$), время работы студентов на УПД ($T_{УПД} \pm \Delta T_{УПД}$), а также процент студентов, желающих работать только на ЭВМ ($P_{ЭВМ}$), процент студентов, желающих работать не только на ЭВМ, но и на УПД ($P_{УПД}$), процент студентов, желающих повторно работать на УПД и ЭВМ ($P_{ПР}$).

Спецификация ограничений на параметры исследуемой системы следующая: исходные данные должны быть положительными числами, процент студентов, желающих работать только на ЭВМ ($P_{ЭВМ}$) и на ЭВМ и УПД ($P_{УПД}$), в сумме должен составлять 100 %, процент студентов, желающих повторно работать на УПД и ЭВМ ($P_{ПР}$), не должен превышать 100 % .

Схема программы (см. рис. 2.8) зависит от выбранного языка моделирования.

Блоки схемы соответствуют блок-диаграмме языка GPSS, что позволит легко написать текст программы, провести ее модификацию и тестирование. Для полного покрытия программы тестами необходимо так подобрать параметры, чтобы все ветви в разветвлениях проходились по меньшей мере по одному разу. Интерпретатор языка GPSS позволяет проанализировать статистические данные по каждой ветви программы.

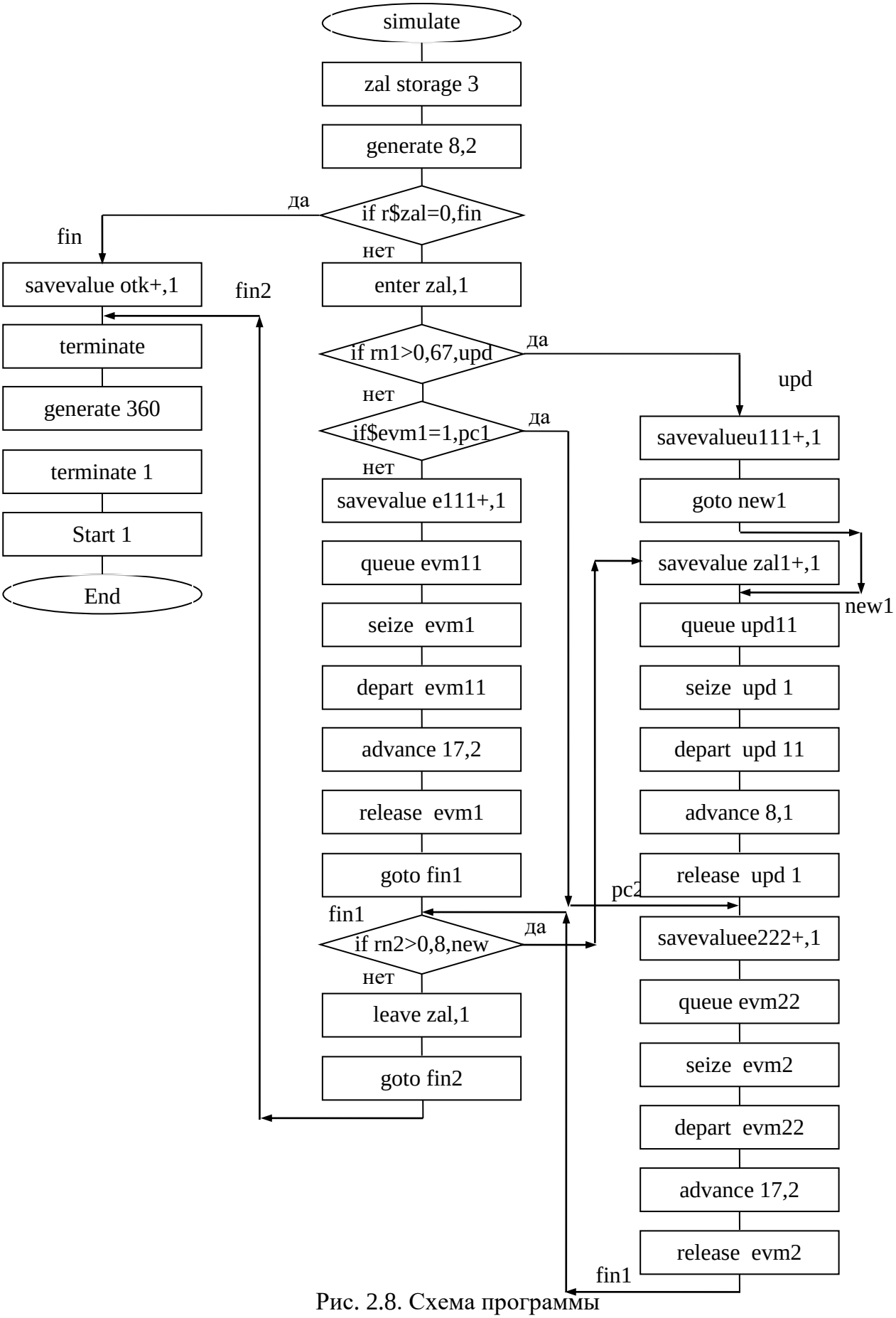


Рис. 2.8. Схема программы

Оценка затрат машинного времени проводится по нескольким критериям эффективности программы: затраты памяти ЭВМ, затраты вычислений (идентичны времени вычислений при последовательной обработке), время вычислений («время ответа»). Данный пункт предлагается проработать самостоятельно. Форма представления входных и выходных данных определяется интерпретатором языка GPSS и изменить ее по усмотрению пользователя невозможно.

2.2.3.2.7. Верификация и проверка достоверности схемы программы

На этом подэтапе проводится верификация программы – доказательство того, что поведение программы соответствует спецификации программы, а также проверка соответствия каждой операции, представленной в схеме программы, аналогичной ей операции в логической схеме модели. Данный пункт предлагается проработать самостоятельно.

2.2.3.2.8. Проведение программирования модели

	Текст программы	Комментарии
	simulate	Начало моделирования
	zal storage 3	Определение емкости многоканального устройства zal – очереди в машинный зал емкостью в 3 заявки
	generate 8,2	Генерация входных заявок
	if r$zal=0,fin	Проверка на занятость устройства zal; если устройство zal занято, то переход по метке fin
	enter zal,1	Занятие устройства zal по одной заявке
	if rn1>0.67,upd	33 % заявок передаются по метке upd
pc1	if f$evm1=1,pc2	Проверка занятости прибора evm1 (ЭВМ 1); если прибор занят, то переход по метке pc2 на прибор evm2
	savevalue e111+,1	Определение количества заявок, которые может обслужить прибор evm1 (ЭВМ 1)
	queue evm11	Сбор статистических данных о входе заявки в очередь evm11 к прибору evm1
	seize evm1	Занятие прибора evm1
	depart evm11	Сбор статистических данных о выходе заявки из очереди evm11 к прибору evm1
	advance 17,2	Обработка заявки в приборе evm1
	release evm1	Освобождение прибора evm1
	goto fin1	Безусловный переход на метку fin1 к выходу из устройства zal
upd	savevalue u111+,1	Определение количества вновь пришедших заявок,

		которые может обслужить прибор upd (УПД)
	goto new1	Безусловный переход на метку new1 для работы на УПД
new	savevalue zal1+,1	Определение количества заявок, которые может повторно обслужить прибор upd
new1	queue upd11	Сбор статистических данных о входе заявки в очередь upd11 к прибору upd
	seize upd	Занятие прибора upd
	depart upd11	Сбор статистических данных о выходе заявки из очереди upd11 к прибору upd
	advance 8,1	Обработка заявки в приборе upd
	release upd	Освобождение прибора upd
pc2	savevalue e222+,1	Определение количества заявок, которые может обслужить прибор evm2
	queue evm22	Сбор статистических данных о входе заявки в очередь evm22 к прибору evm2
	seize evm2	Занятие прибора evm2
	depart evm22	Сбор статистических данных о выходе заявки из очереди evm22 к прибору evm2
	advance 17,2	Обработка заявки в приборе evm2
	release evm2	Освобождение прибора evm2
fin1	if rn2>0.8,new	20 % заявок передается по метке new для повторной обработки на приборах upd и evm2
	leave zal,1	Освобождение устройства zal по одной заявке
	goto fin2	Безусловный переход на метку fin2 для уничтожения заявок
fin	savevalue otk+,1	Определение количества отказов в обслуживании заявок в устройстве zal
fin2	terminate	Уничтожение заявок
	generate 360	Генерация временных заявок для моделирования 60 часов работы
	terminate 1	Уничтожение временных заявок
	start 1	Продолжение работы системы
	end	Окончание моделирования

2.2.3.2.9. Проверка достоверности программы

На данном подэтапе последняя проверка машинной реализации модели проводится следующим образом:

а) обратным переводом программы в исходную схему;

б) проверкой отдельных частей программы при решении различных тестовых задач;

в) объединением всех частей программы и проверкой ее в целом на контрольном примере моделирования варианта системы.

На этом подэтапе необходимо также проверить затраты машинного времени на моделирование. Данный пункт предлагается проработать самостоятельно.

2.2.3.3. Получение и интерпретация результатов моделирования системы

2.2.3.3.1. Планирование машинного эксперимента с моделью системы

Для получения максимального объема необходимой информации об объекте моделирования при минимальных затратах машинных ресурсов проведем полный факторный эксперимент с четырьмя существенными факторами (переменных и параметров).

Согласно выбранным критериям оценки эффективности системы и целевой функции модели выберем следующие существенные факторы:

x_1 – допустимая очередь в машинный зал, $L_{ЗАЛ} = 3$;

x_2 – интервал времени прихода студентов в зал, $T_{ПР} = 8$ мин;

x_3 – время работы студентов на ЭВМ1, $T_{ЭВМ1} = 17$ мин;

x_4 – время работы студентов на ЭВМ2, $T_{ЭВМ2} = 17$ мин.

Зададим уровни вариации для каждого фактора:

$\Delta x_1 = 1$, $\Delta x_2 = 4$, $\Delta x_3 = 7$, $\Delta x_2 = 10$.

Составим матрицу плана полного факторного эксперимента.

Номер опыта	Фактор x_1	Фактор x_2	Фактор x_3	Фактор x_4
0 (базовый)	3	8	17	17
1	2	4	10	7
2	2	4	10	27
3	2	4	24	7
4	2	4	24	27
5	2	12	10	7
6	2	12	10	27

Номер опыта	Фактор x_1	Фактор x_2	Фактор x_3	Фактор x_4
7	2	12	24	7
8	2	12	24	27
9	4	4	10	7
10	4	4	10	27
11	4	4	24	7
12	4	4	24	27
13	4	12	10	7
14	4	12	10	27
15	4	12	24	7
16	4	12	24	27

2.2.3.3.2. Определение требований к вычислительным средствам

Для проведения эксперимента потребуется только один персональный компьютер без внешних устройств. Время выполнения эксперимента ограничено лишь временем доступа к персональному компьютеру.

2.2.3.3.3. Проведение рабочих расчетов

Набор исходных данных для ввода в ЭВМ представлен в виде матрицы плана, с помощью которой в достаточном объеме исследуется факторное пространство. Получение выходных данных зависит от интерпретатора языка GPSS. Дополнительные расчеты не требуются.

2.2.3.3.4. Анализ результатов моделирования системы

Планирование полного факторного эксперимента с моделью позволяет вывести необходимое количество выходных данных, при этом каждый опыт соответствует одному из возможных состояний исследуемой системы. Статистические характеристики модели вычисляются в интерпретаторе языка GPSS автоматически. Проведение регрессионного, корреляционного и дисперсионного анализа не требуется.

2.2.3.3.5. Представление результатов моделирования

Результаты моделирования представлены в табл. 2.4, 2.5, 2.6. Относительное время – 360, абсолютное время – 360 единиц времени (мин). Выходная переменная e111 соответствует количеству заявок, которые может обслужить прибор evm1 (ЭВМ 1). Выходная переменная u111 соответствует количеству вновь пришедших заявок, которые может обслужить прибор upd (УПД). Выходная переменная zal1 соответствует количеству заявок, которые может повторно обслужить прибор upd. Выходная переменная e222 соответствует количеству заявок, которые может обслужить прибор evm2 (ЭВМ 2). Выходная переменная otk соответствует количеству заявок, которым было отказано в обслуживании в устройстве zal.

Таблица 2.4

Результаты работы устройств evm1, upd, evm2

Номер	Устройство	Средняя загрузка	Число входов	Среднее время транзакции	Значение переменной				
					e111	u111	zal1	e222	otk
1	2	3	4	5	6	7	8	9	10
0	evm1	0,40	9	15,80	9	10	7	21	21
	upd	0,37	17	7,76					
	evm2	0,93	20	16,80					
1	evm1	0,46	17	9,83	17	15	9	30	45
	upd	0,52	24	7,81					
	evm2	0,59	30	7,03					
2	evm1	0,52	19	9,76	19	5	5	13	64
	upd	0,22	10	7,93					
	evm2	0,86	12	25,79					
3	evm1	0,78	12	23,53	12	9	10	27	57
	upd	0,43	19	8,11					
	evm2	0,50	27	6,71					
4	evm1	0,60	9	24,11	9	6	2	12	69
	upd	0,18	8	8,17					
	evm2	0,87	12	25,96					
5	evm1	0,34	12	10,24	12	12	5	18	4
	upd	0,39	17	8,18					
	evm2	0,34	18	6,79					
6	evm1	0,36	13	9,97	13	6	5	11	10
	upd	0,24	11	7,84					
	evm2	0,75	10	26,82					
7	evm1	0,55	9	22,08	9	8	6	22	4
	upd	0,31	14	8,07					
	evm2	0,43	22	7,07					
8	evm1	0,48	8	21,82	8	7	3	10	14
	upd	0,23	10	8,13					
	evm2	0,74	10	26,78					
9	evm1	0,55	21	9,42	21	24	11	52	26
	upd	0,74	35	7,60					
	evm2	0,97	51	6,87					
10	evm1	0,29	10	10,42	10	5	7	16	76
	upd	0,26	12	7,93					
	evm2	0,97	13	26,81					

Окончание табл. 2.4

1	2	3	4	5	6	7	8	9	10
11	evm1	0,77	12	23,01	12	22	11	52	34
	upd	0,74	33	8,06					
	evm2	0,95	51	6,70					
12	evm1	0,52	8	23,58	8	8	5	15	69
	upd	0,29	13	7,92					
	evm2	0,96	13	26,66					
13	evm1	0,45	16	10,17	16	11	8	18	0
	upd	0,39	18	7,77					
	evm2	0,35	18	7,02					
14	evm1	0,62	22	10,11	22	4	5	12	0
	upd	0,20	9	7,96					
	evm2	0,83	11	27,11					
15	evm1	0,73	11	23,73	11	8	7	25	0
	upd	0,33	15	7,88					
	evm2	0,47	24	7,02					
16	evm1	0,59	9	23,56	9	8	3	15	8
	upd	0,25	11	8,03					
	evm2	0,93	13	25,78					

Таблица 2.5

Результаты работы накопителя zal

Номер	Накопитель	Емкость	Среднее значение	Средняя загрузка	Число входов	Среднее время транзакции	Текущее значение	Максимальное значение
1	2	3	4	5	6	7	8	9
0	zal	3	2,56	0,85	23	40,14	3	3
1	zal	2	1,64	0,82	39	15,11	1	2
2	zal	2	1,79	0,89	27	23,83	2	2
3	zal	2	1,75	0,88	29	21,78	2	2
4	zal	2	1,87	0,93	19	35,36	2	2
5	zal	2	1,12	0,56	25	16,15	1	2
6	zal	2	1,48	0,74	19	28,12	2	2
7	zal	2	1,33	0,66	25	19,13	1	2
8	zal	2	1,64	0,82	15	39,30	1	2
9	zal	4	3,34	0,84	63	19,11	4	4
10	zal	4	3,78	0,94	19	71,62	4	4
11	zal	4	3,50	0,88	53	23,78	3	4
12	zal	4	3,79	0,95	18	75,74	3	4
13	zal	4	1,26	0,32	28	16,20	1	3
14	zal	4	2,16	0,54	29	26,79	3	4

Окончание табл. 2.5

1	2	3	4	5	6	7	8	9
15	zal	4	1,61	0,40	29	19,97	3	4
16	zal	4	3,31	0,83	21	56,80	4	4

Таблица 2.6

Результаты работы очередей evm11, upd11, evm22

Номер	Очередь	Максимальное значение	Среднее значение	Всего входов	Ноль входов	Процент нолей	Среднее время транзакции	Среднее время транзак-ции, кроме нулевых входов	Текущее значение
1	2	3	4	5	6	7	8	9	10
0	evm11	1	0,00	9	9	100,00	0,00	0,00	0
	upd11	1	0,02	17	15	88,24	0,42	3,58	0
	evm22	2	0,85	21	3	14,29	14,56	16,98	1
1	evm11	1	0,00	17	17	100,00	0,00	0,00	0
	upd11	1	0,06	24	15	62,67	0,95	2,53	0
	evm22	1	0,00	30	29	96,67	0,05	1,39	0
2	evm11	1	0,00	19	19	100,00	0,00	0,00	0
	upd11	1	0,00	10	10	100,00	0,00	0,00	0
	evm22	1	0,19	13	8	61,54	5,31	13,81	1
3	evm11	1	0,00	12	12	100,00	0,00	0,00	0
	upd11	1	0,04	19	16	84,21	0,74	4,66	0
	evm22	1	0,00	27	27	100,00	0,00	0,00	0
4	evm11	1	0,00	9	9	100,00	0,00	0,00	0
	upd11	1	0,02	8	6	75,00	1,10	4,41	0
	evm22	1	0,19	12	8	66,67	5,76	17,27	0
5	evm11	1	0,00	12	12	100,00	0,00	0,00	0
	upd11	1	0,05	17	12	70,59	1,12	3,80	0
	evm22	1	0,00	18	17	94,44	0,02	0,44	0
6	evm11	1	0,00	13	13	100,00	0,00	0,00	0
	upd11	1	0,00	11	11	100,00	0,00	0,00	0
	evm22	1	0,14	11	5	45,45	4,57	8,38	1
7	evm11	1	0,00	9	9	100,00	0,00	0,00	0
	upd11	1	0,03	14	11	78,57	0,78	3,65	0
	evm22	1	0,00	22	22	100,00	0,00	0,00	0
8	evm11	1	0,00	8	8	100,00	0,00	0,00	0
	upd11	1	0,00	10	10	100,00	0,00	0,00	0
	evm22	1	0,18	10	4	40,00	6,58	10,97	0

Окончание табл. 2.6

1	2	3	4	5	6	7	8	9	10
9	evm11	1	0,00	21	21	100,00	0,00	0,00	0
	upd11	2	0,24	35	14	40,00	2,48	4,13	0
	evm22	2	0,84	52	2	3,85	5,82	6,05	1
10	evm11	1	0,00	10	10	100,00	0,00	0,00	0
	upd11	1	0,00	12	12	100,00	0,00	0,00	0
	evm22	3	2,26	16	1	6,25	50,81	54,20	3
11	evm11	1	0,00	12	12	100,00	0,00	0,00	0
	upd11	2	0,35	33	12	36,36	3,78	5,94	0
	evm22	2	0,70	25	8	15,38	4,85	5,73	1
12	evm11	1	0,00	8	8	100,00	0,00	0,00	0
	upd11	1	0,02	13	10	76,92	0,65	2,80	0
	evm22	3	1,99	15	1	6,67	47,78	51,19	2
13	evm11	1	0,00	16	16	100,00	0,00	0,00	0
	upd11	1	0,06	18	11	61,11	1,19	3,05	0
	evm22	1	0,00	18	16	88,89	0,19	1,71	0
14	evm11	1	0,00	22	22	100,00	0,00	0,00	0
	upd11	1	0,00	9	9	100,00	0,00	0,00	0
	evm22	2	0,51	12	3	25,00	15,39	20,53	1
15	evm11	1	0,00	11	11	100,00	0,00	0,00	0
	upd11	1	0,03	15	12	80,00	0,84	4,18	0
	evm22	2	0,05	25	19	76,00	0,76	3,17	1
16	evm11	1	0,00	9	9	100,00	0,00	0,00	0
	upd11	1	0,0	11	10	90,91	0,46	5,05	0
	evm22	3	1,53	15	1	6,67	36,82	39,45	2

2.2.3.3.6. Интерпретация результатов моделирования

Полученные результаты можно интерпретировать следующим образом.

Согласно целевой функции оптимальными вариантами модели являются опыты № 13, 14, 15, т.к. количество отказов равно 0. ЭВМ1, УПД и ЭВМ2 загружены равномерно (0,45; 0,39; 0,35 % соответственно), обработано максимальное количество заявок. Это объясняется тем, что студенты приходят в машинный зал (фактор x_2) реже – с интервалом 12 ± 2 мин; допустимая очередь в машинный зал (фактор x_1) увеличена и составляет пять человек, включая работающего на УПД; работа на ЭВМ1 (фактор x_3) уменьшена и занимает 10 ± 1 мин (для опытов № 13, 14) и 24 ± 1 мин (для опыта № 15); работа на ЭВМ2 (фактор x_4) занимает 7 ± 1 мин (опыты № 13, 15) и 27 ± 1 мин (опыт № 14). Самым лучшим опытом является опыт №13: ни одной заявки в очереди к ЭВМ1, УПД и ЭВМ2 не осталось, поскольку время работы на ЭВМ1 и ЭВМ2 меньше, чем время прихода студентов в машинный зал, и все студенты, в т.ч. и повторно желающие поработать на УПД и ЭВМ, успевают обслужиться за общее время моделирования.

Наихудшими вариантами модели являются опыты № 4, 10, 12, т.к. количество отказов максимально (69, 76, 69 соответственно), ЭВМ1, УПД и ЭВМ2 загружены неравномерно (опыт № 4 – 0,60; 0,18; 0,87 % соответственно, опыт № 10 – 0,29; 0,26; 0,97 % соответственно, опыт № 12 – 0,52; 0,29; 0,96 % соответственно), обработано минимальное количество заявок (21, 26, 23 соответственно). Это объясняется тем, что студенты приходят в машинный зал (фактор x_2) часто с интервалом 4 ± 2 мин, допустимая очередь в машинный зал (фактор x_1) составляет 3 (для опыта № 4) и 5 (для опытов № 10, 12) человек, включая работающего на УПД, работа на ЭВМ1 (фактор x_3) занимает 24 ± 1 мин (для опытов № 4, 12) и 10 ± 1 мин (для опыта № 10), работа на ЭВМ2 (фактор x_4) увеличена и занимает 27 ± 1 мин. Самым худшим опытом является опыт №10: количество отказов максимально – 76, в очереди к ЭВМ2 осталось 3 заявки, поскольку время работы на ЭВМ1 и ЭВМ2 больше. При этом загрузка ЭВМ2 значительно больше загрузки ЭВМ1, т.к. в данной модели все студенты, повторно желающие поработать на УПД, обязательно идут на ЭВМ2. Поэтому лучше было реализовать модель, в которой студенты после работы на УПД занимали бы свободную ЭВМ.

2.2.3.3.7. Подведение итогов моделирования и выдача рекомендаций

Результаты моделирования при проведении машинного эксперимента подтвердили следующие гипотезы для базовой точки эксперимента:

- загрузка УПД будет меньше загрузки ЭВМ;
- вероятность отказа в обслуживании вследствие переполнения очереди к УПД или ЭВМ больше 0,6 %, т.к. 30 заявок обслужено, 21 – получила отказ в обслуживании;
- соотношение желающих работать на ЭВМ и УПД в очереди составило практически 2:1 (29 заявок обработано на ЭВМ и 17 на УПД).

Рекомендации по практическому использованию результатов моделирования предлагается проработать самостоятельно.

2.2.4. Задания для самостоятельной работы

Задание 1

В вычислительный зал с интервалом времени 10 ± 5 мин заходят пользователи, желающие произвести расчеты на ЭВМ. В зале имеется 8 ЭВМ, к одной ЭВМ подключен принтер. Время, необходимое для решения задачи и вывода результатов на печать, составляет 15 ± 5 мин. Вывод результатов на печать не мешает проведению расчетов, время печати составляет 3 ± 2 мин. В зал не допускается более 10 студентов.

Смоделировать процесс обслуживания 100 пользователей. Подсчитать число пользователей, не нашедших свободного места в очереди. Определить среднее число пользователей в очереди, а также коэффициенты загрузки каждой ЭВМ и принтера. Составить план полного факторного эксперимента

для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 2

Моделирование вычислительной системы ВС конвейерного типа (рис. 2.9). Данная система состоит из трех процессоров и оперативной памяти. Операнды проходят последовательную обработку в процессорах под воздействием команд, причем для некоторых процессоров существуют так называемые «пустые» команды. Список условных кодов команд с их временными характеристиками приведен в табл. 2.7.

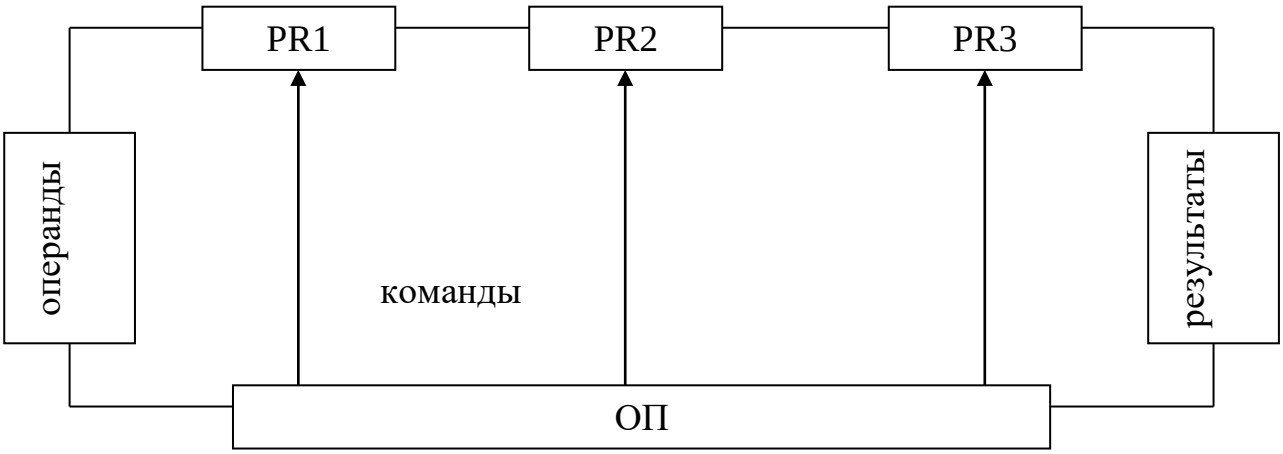


Рис. 2.9. Вычислительная система конвейерного типа

Программа для первого процессора PR1 записана в карте функции КОММ1. Здесь аргументами функции являются значения времени, в которые должна выполняться та или иная команда, а в результате будет выдаваться код команды. Для второго и третьего процессоров существуют функции КОММ2 и КОММ3 соответственно. Функция TIME1 описывает среднее время выполнения команд, а TIME2 – половину поля равномерного распределения команд. Переменная CI содержит номер выполняемой команды в программе.

Таблица 2.7

Код команды	Время задержки
0	2±1
1	3±2
2	4±1
3	4±1
4	3±1

Смоделировать процесс обслуживания 100 операндов. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени

характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 3

Система обработки и хранения информации (рис. 2.10) состоит из центрального процессора (ЦП) с ограниченным набором обрабатываемой информации и вспомогательной шины, состоящей из двух параллельно работающих процессоров (ВП1 и ВП2), которые подключаются при загрузке центрального процессора. Устройство управления (УУ) управляет работой ЦП, ВП1 и ВП2. Вся обработанная информация поступает на устройство сбора информации (УСИ), откуда пакетами по n заявок поступает в блок хранения информации (БХИ). Количество пользователей (П1, П2 и П3) ограничено тремя.

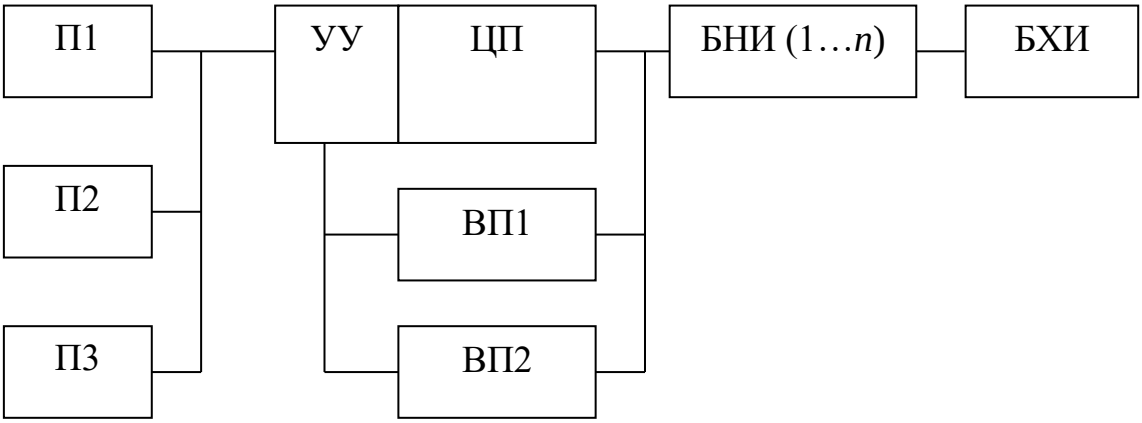


Рис. 2.10. Система обработки и хранения информации

Смоделировать процесс обслуживания 100 пользователей. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 4

Вычислительная система содержит: 3 терминала, 2 буферных накопителя, 2 ЭВМ (см. рис. 2.11). Пользователи приходят через интервалы времени 10 ± 2 мин. Если все три имеющихся терминала заняты, пользователю отказывают в обслуживании. Терминалы могут обеспечить обслуживание средней программы пользователя за 20 ± 5, 40 ± 10 и 40 ± 20 мин соответственно номерам терминалов. Полученные программы сдаются в буферные накопители, откуда выбираются на обработку на первую ЭВМ – программы с 1-го и 2-го терминалов, на 2-ю ЭВМ – программы с 3-го терминала. Время обработки программ на 1-й и 2-й ЭВМ равно 15 и 30 соответственно.

Смоделировать процесс обслуживания 100 пользователей. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей

степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

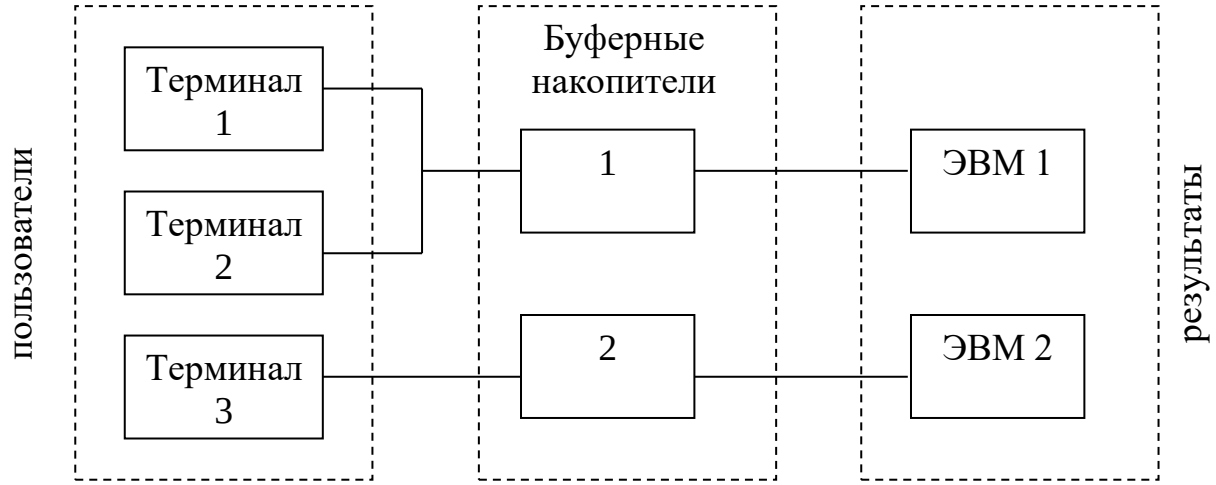


Рис. 2.11. Вычислительная система массового обслуживания

Задание 5

В процессор Пр поступают заявки от внешнего устройства ВУ1, при этом заполняется ОЗУ и процессор работает. Если в ОЗУ остается свободное место, то еще поступают заявки от ВУ2, иначе процессор работает с одной заявкой (рис. 2.12). Если от ВУ поступила заявка больше 640 кБ, то она не обслуживается, т.е. происходит переполнение ОЗУ.

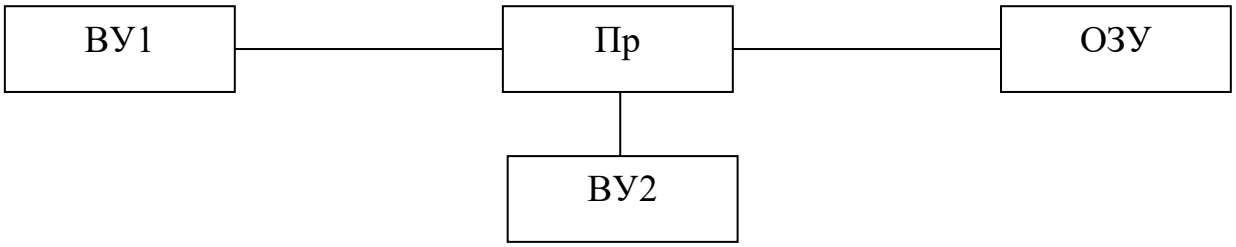


Рис. 2.12. Вычислительная система

Смоделировать процесс обслуживания в течение 8 ч. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 6

В состав двухпроцессорной системы входят: два процессора под символическими именами PR1 и PR2 и оперативная память (см. рис. 2.13). Система обрабатывает поток заявок Z1 и Z2, поступающих в систему по равномерному закону с интервалами 4 ± 2 и 25 ± 2 соответственно. Поток заявок Z2 обрабатывается только на PR1 и придерживает обработку заявок потока Z1, после чего прерванная заявка дообслуживается. Обслуживание заявок потока Z1 происходит по условию: если очередь PR2 > PR1, то

обслуживание происходит на PR1, иначе на PR2. После обработки на процессорах заявка помещается в оперативную память.

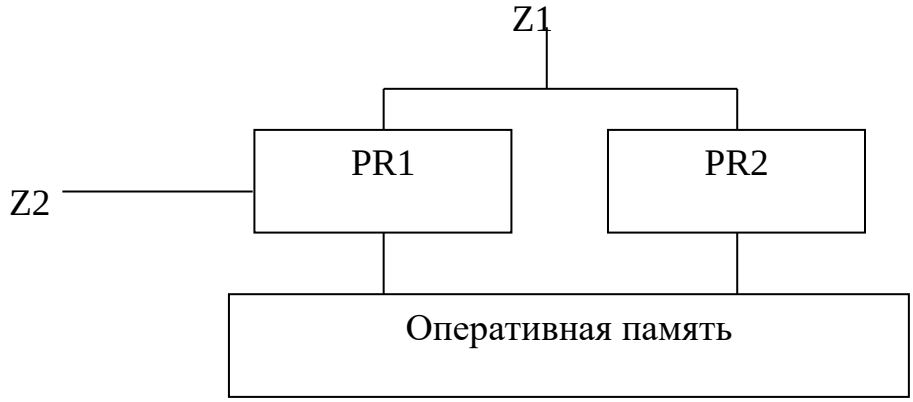


Рис. 2.13. Двухпроцессорная система

Смоделировать процесс обслуживания в течение 10 ч. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 7

Имеются три независимо работающих процессора с разными временами задержки и устройство неограниченной по размерам внешней памяти (рис. 2.14). Поток заявок, поступающий с периферийных устройств ПУ, обрабатывается на том процессоре, очередь к которому меньше. После обработки заявки происходит захват памяти, куда передаются результаты обработки.

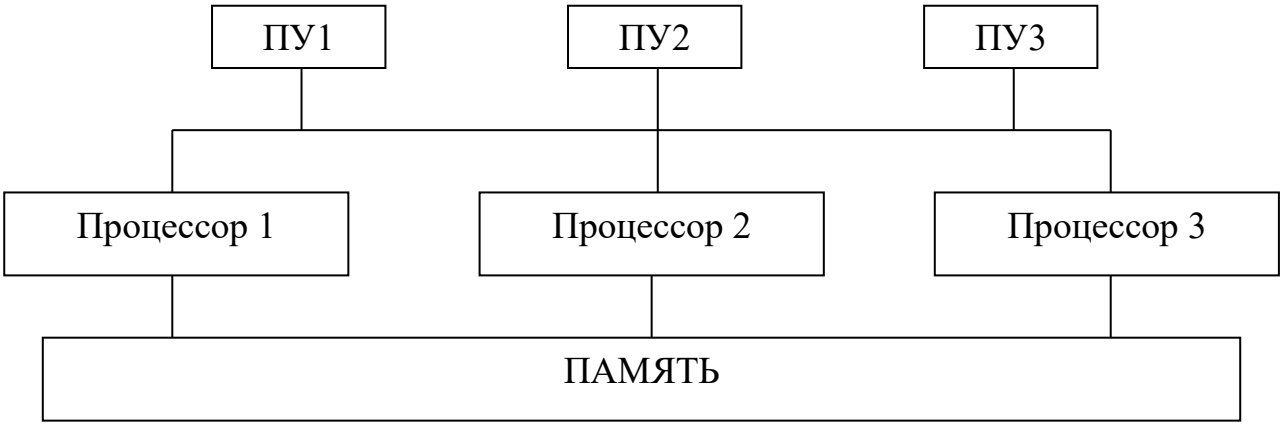


Рис. 2.14. Распределенная вычислительная система

Смоделировать процесс обслуживания в течение 12 ч. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 8

В вычислительном классе на обработку принимаются три класса заданий: A, B, C. Исходя из наличия оперативной памяти ЭВМ, задания классов A и B могут решаться одновременно, а задания класса C монополизируют ЭВМ. Задания класса A поступают через 20 ± 5 мин, класса B – через 20 ± 10 мин и класса C – через 30 ± 10 мин и требуют для выполнения: класс A – 20 ± 5 мин, класс B – 21 ± 3 мин и класс C – 28 ± 5 мин. Задачи класса C загружаются в ЭВМ, если она полностью свободна. Задачи класса A и B могут дозагружаться к решающей задаче. Смоделировать процесс обслуживания в течение 24 ч. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 9

Информационно-поисковая библиографическая система построена на базе двух ЭВМ и имеет один терминал для ввода и вывода информации. Первая ЭВМ обеспечивает поиск научно-технической литературы (вероятность обращения к ней – 0,7), а вторая – медицинской (вероятность обращения к ней – 0,3). Пользователи обращаются к услугам системы каждые 5 ± 2 мин. Если в очереди к терминалу ожидают 10 пользователей, то вновь прибывшие пользователи получают отказ в обслуживании. Поиск информации на первой ЭВМ продолжается 6 ± 4 мин, а на второй 3 ± 2 мин. Для установления связи с нужной ЭВМ и передачи текста запроса пользователи тратят 2 ± 1 мин. Вывод результатов поиска происходит за 1 мин. Смоделировать процесс работы системы за 8 ч. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

Задание 10

Пять операторов работают в справочной телефонной сети города, сообщая номера телефонов по запросам абонентов, которые обращаются по одному номеру – 09. Автоматический коммутатор переключает абонента на того оператора, в очереди которого ожидает наименьшее количество абонентов, причем наибольшая допустимая длина очереди перед оператором – два абонента. Если все очереди имеют максимальную длину, вновь поступивший вызов получает отказ. Обслуживание абонентов операторами длится 30 ± 20 с. Вызовы поступают в справочную службу через каждые 5 ± 3 с. Смоделировать обслуживание 100 вызовов. Проанализировать характеристики СМО. Составить план полного факторного эксперимента для четырех существенных факторов, которые в наибольшей степени характеризуют свойства системы, и исследовать поведение модели СМО, меняя параметры системы.

2.3. Моделирование систем управления с помощью пакета VISSIM

2.3.1. Описание пакета программ VISSIM

Пакет программ VISSIM предназначен для моделирования процессов в системах управления, которые можно описать дифференциальными уравнениями. Работа пакета базируется на интегрировании дифференциальных уравнений, которые вводятся в пакет с помощью графических символов – блоков структурных схем с соответствующими значениями параметров. Кроме того, в данном пакете имеется библиотека вспомогательных блоков, позволяющих полно анализировать динамические системы управления.

Технические требования к пакету: операционная система не ниже Microsoft® Windows 3.1, 2 МБ оперативной памяти, 1 МБ на жестком диске, мышь, монитор с разрешающей способностью не ниже 800x600.

Для запуска программы необходимо перейти в каталог, где установлена программа, и двойным щелчком левой клавиши (ЛК) мыши нажать на иконку с подписью PVISSIM.EXE. Возможен и другой вариант запуска программы, которую предоставляет операционная система. После этого вы увидите главное окно (рис. 2.15).

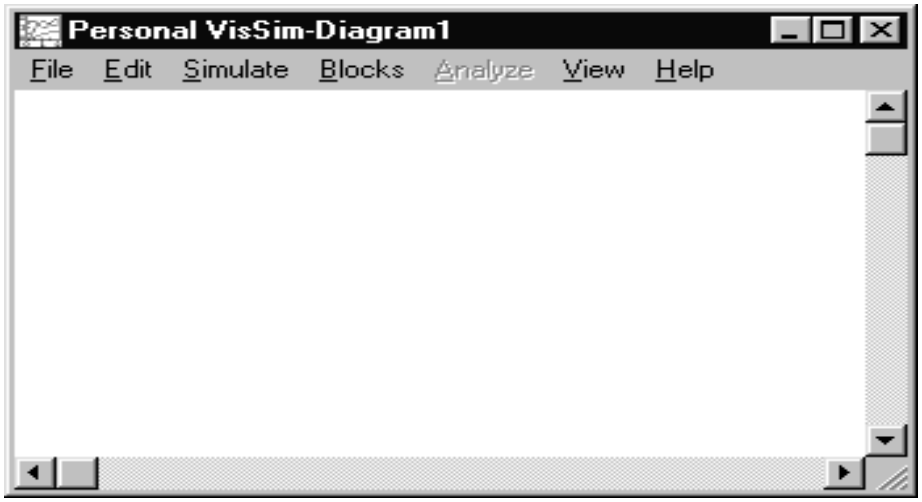


Рис. 2.15. Главное окно пакета программ VISSIM

Перед началом работы с пакетом вы можете настроить вид рабочей области так, как приятно вашему глазу. Подробно опишем, как это делать.

2.3.1.1. Изменение параметров рабочей области

Для изменения параметров рабочей области предусмотрено меню View (рис. 2.16), состоящее из трех элементов Fonts…, Colors…, Presentation.

Элемент меню Fonts (рис. 2.17) предназначен для изменения параметров отображаемого шрифта: гарнитуры, стиля, размера, цвета. Также имеется поле для предварительного просмотра, отображаемого шрифта.

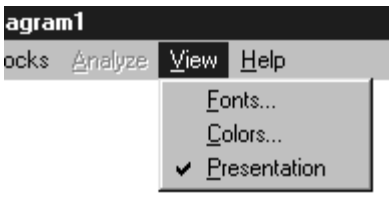


Рис. 2.16. Меню View

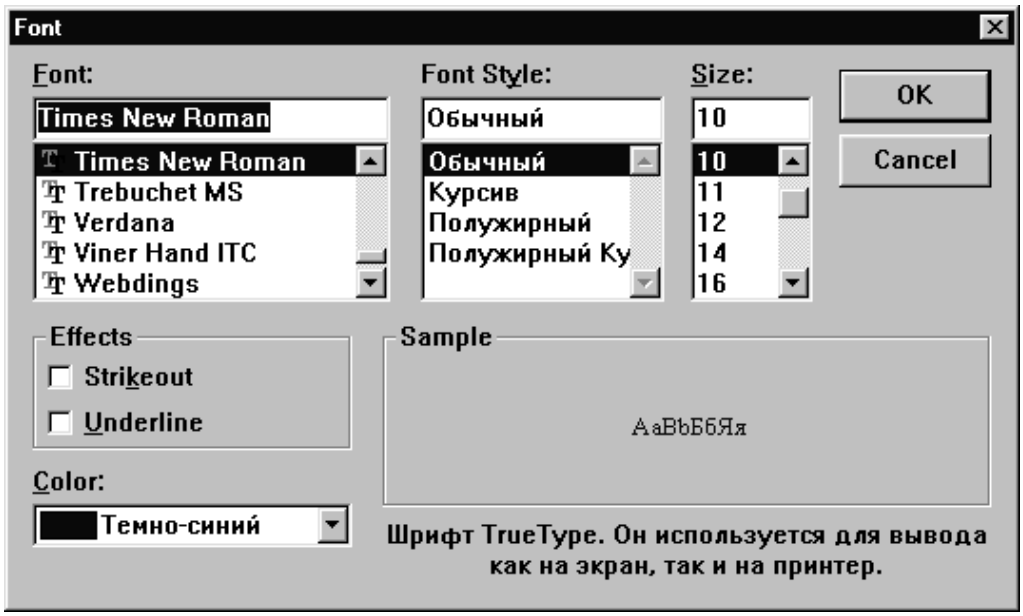


Рис. 2.17. Диалог Font

Элемент меню Colors (рис. 2.18) предназначен для изменения цвета фона рабочей области (window background), цвета фона графиков (plot background), цвета соединительных линий (wires), цвета текста элементов (text).

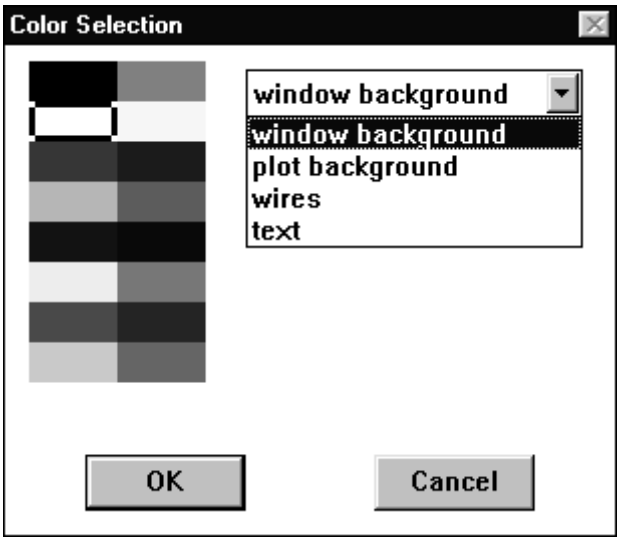


Рис. 2.18. Диалог Color

Элемент меню Presentation управляет видом стрелок на соединительных линиях: если рядом с ним стоит галочка, то стрелки становятся закрашенными и тоненькими, иначе – прозрачными и широкими.

После того как вы определились со стилем (желательно задать белый цвет фона рабочей области и графиков), вам необходимо в рабочем поле изображать структурную схему исследуемой системы, состоящей из стандартных блоков и связей между ними.

2.3.1.2. Стандартные блоки

Пакет VISSIM предлагает богатый набор стандартных блоков:

- annotation (блоки аннотаций);
- arithmetic (арифметические блоки);
- boolean (логические блоки);
- integration (блоки интегрирования);
- nonlinear (блоки нелинейности);
- random generator (генераторы случайных чисел);
- real time (блоки задания времени);
- signal consumer (блоки-приемники сигналов);
- signal producer (блоки задания сигналов);
- time delay (блоки задержки времени);
- transcendental (блоки трансцендентных функций);
- DDE (блок динамического обмена);
- userFunction (блок собственной функции-программы);
- neuralNet (блок для обмена данными по сети).

Также имеется ряд вспомогательных блоков для организации более легко воспринимаемых глазом структурных схем.

Для вставки блока в рабочую область необходимо вызвать меню Blocks (см. рис. 2.19), выбрать нужный элемент и, нажав ЛК мыши, перетащить блок в рабочую область. Для соединения блоков необходимо, нажав ЛК мыши, соединить выход одного блока с входом другого. Для удаления связи необходимо подвести курсор мыши к концу связи (вход второго блока) и, нажав ЛК мыши, отвести связь в пустое пространство рабочей области. Большинство блоков имеют параметры; для их изменения необходимо нажать правую кнопку (ПК) мыши на интересующем нас блоке и изменить необходимые параметры.

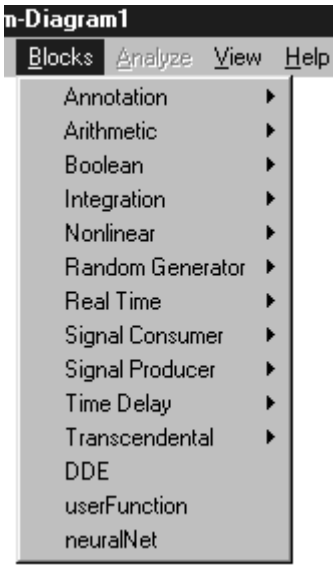


Рис. 2.19. Меню Blocks

Далее описаны функции блоков, которые предоставляет изучаемый пакет программ. Отметим, что подменю Real Time, neuralNet, DDE, userFunction требуют дополнительно установленных компонент пакета программ.

2.3.1.2.1. Блоки меню Annotation (аннотаций)

Comment	Вставляет комментарий в рабочую область, в нем можно писать что угодно, так как это не повлияет на расчет системы.
Date	Вставляет текущую дату и время в формате: день недели, месяц, число, время, год. К сожалению, формат изменить нельзя.
ScalarToVec	Преобразует скалярные данные на входе в один вектор. Можно использовать в больших схемах, в которых много параметров, для более зрительного восприятия. По умолчанию три входа. Можно добавить необходимое количество входов с помощью команды Add Input меню Edit.
VecToScalar	Обратное действие ScalarToVec. По умолчанию три выхода. Можно добавить необходимое количество входов с помощью команды Add Output меню Edit.
Variable	Через этот блок можно данные из одной части диаграммы передать в другую, не используя соединительные линии. Все блоки подобного типа, но с одинаковым именем, считаются одинаковыми, однако только один блок может иметь соединительную линию на входе. Если имя блока начинается с символа «:» (двоеточие), то такой блок считается локальным, т.е. доступен только в текущем блоке. Существуют четыре встроенных имени блока: - $firstPass – генерирует сигнал прямоугольной формы, если идет первый шаг модельного времени;

- $lastPass – генерирует сигнал прямоугольной формы, если идет последний шаг модельного времени;
- $runCount – указывает число проведенных экспериментов (отсчет от 1);
- $timeStep – возвращает шаг моделирования.

WireLabel	Вставляет надпись в рабочую область. Очень удобно для подписи соединительных линий.
WirePositioner	Вставляет стрелку. Обычно блоки уже имеют входные и выходные стрелки, но иногда соединение, проводимое при помощи имеющихся стрелок, портит вид схемы. Используют этот элемент для исправления этой проблемы.

2.3.1.2.2. Блоки меню Arithmetic (арифметические)

1/X	На выходе число, обратно пропорциональное числу на входе.
-X	На выходе число, противоположное по знаку числу на входе.
*	Умножитель, число множителей не ограничено.
/	Делит два числа: верхнее на нижнее.
Abs	Возвращает модуль числа.
Gain	Умножение на константу.
Pow	Возводит число в указанную степень.
Sign	Возвращает знак числа: 1 – положительное, -1 – отрицательное, 0 – ноль.
Summing-Function	Сумматор. Число входов не ограничено.

2.3.1.2.3. Блоки меню Boolean (логические)

>	Возвращает 1, если верхнее число больше нижнего, иначе 0.
<	Возвращает 1, если верхнее число меньше нижнего, иначе 0.
>=	Возвращает 1, если верхнее число не меньше нижнего, иначе 0.
<=	Возвращает 1, если верхнее число не больше нижнего, иначе 0.
==	Возвращает 1, если числа равны, иначе 0.
!=	Возвращает 1, если числа не равны, иначе 0.
Not	Логическое отрицание.
And	Логическое умножение (по битам).
Or	Логическое сложение.
Xor	Сложение по модулю 2 («исключающее или»).

2.3.1.2.4. Блоки меню Integration (интегрирование)

Integrator	Численное интегрирование
Limited Integrator	Определенный интеграл
Reset Integrator	Интегратор со сбросом
Transfer Function	Задает передаточную функцию

2.3.1.2.5. Блоки меню Nonlinear (нелинейные)

CrossDetect	Определяет прохождение сигнала через заданную точку, причем на выходе звена 1 – если идет нарастание сигнала, -1 – если убывание, 0 – если сигнал не пересекает данную точку
DeadBand	Задает зону нечувствительности к входному воздействию
Int	Усекает число с плавающей точкой до целого
Limit	Ограничитель: сигнал не может выходить за указанный диапазон
Map	Позволяет в соответствии с входными воздействиями определить значение выхода. Соответствие ставится по следующему закону: зависимые переменные интерполируются между точками независимой переменной и экстраполируются за границами изменения независимой переменной. Возможно два типа сопоставления. Первый тип – одной входной переменной соответствует несколько зависимых переменных. Второй тип – двум входным переменным соответствует одна зависимая переменная
Max	Возвращает максимальное из двух чисел
Min	Возвращает минимальное из двух чисел
Merge	Мультиплексирование двух сигналов
Quantize	Округляет с избытком или недостатком входную величину в соответствии с параметром
Relay	Описывает трех-, двухпозиционное реле
SampleHold	Описывает защелку. Если управляющий сигнал не меньше 1, то изменяется внутреннее значение и выход, иначе ничего не изменяется.

2.3.1.2.6. Блоки меню Random Generator (генераторы случайных чисел)

Gaussian	Возвращает нормально распределенные случайные числа с заданными математическим ожиданием (МО) и среднеквадратическим отклонением (СКО)
Uniform	Возвращает равномерно распределенные случайные числа в диапазоне (0,1)

2.3.1.2.7. Блоки меню Signal Consumer (приемники сигналов)

Constraint	Задает точность для итерационных расчетов, присоединяется там, где выражение принимает нулевое значение. Используется вместе с блоком unknown из меню Signal Producer
Display	Выводит текущее значение сигнала
Error	Останавливает моделирование, если на входе не нуль. Блок, который остановил моделирование, подсвечивается красным цветом.
Export	Запись сигналов в текстовый файл
Meter	Интерактивная форма аналогового вольтметра
Plot	На входы этого блока подаются сигналы, которые отображаются во временной области на графике в ходе моделирования.
Stop	Останавливает моделирование. Если моделирование проводилось с автоперезапуском параметров и значение на входе не меньше двух, то происходит полная остановка; если значение на входе не меньше единицы, то прекращается текущее моделирование и приступают к новому шагу моделирования, возможно, с другими параметрами.

2.3.1.2.8. Блоки меню Signal Producer (генераторы сигналов)

Button	Кнопка. Если темная, то – 1, иначе – 0
Const	Константа
Import	Генерирование сигнала на основе данных из файла
Parabola	Возвращает значение квадратичного трехчлена. В качестве независимой переменной выступает модельное время
PulseTrain	Генерирует периодические всплески амплитуды сигнала. Используются для тактирования элементов задержки
Ramp	Возвращает текущее модельное время (блок линейно нарастающего сигнала)
RealTime	Возвращает время (в миллисекундах), прошедшее со времени старта программы
Sinusoid	Генерирует сигнал синусоидальной формы
Slider	Блок задает константу, которую можно менять во время моделирования
Step	Задает единичную функцию
Unknown	Выход этого блока задает неизвестную величину в

	статически заданных (алгебраических) уравнениях, а вход – начальные условия для итерационного процесса

2.3.1.3. Меню File

Меню File содержит стандартные для Windows функции, необходимые для работы с файлом в рабочей области пакета.

New	Очищает рабочую область. Если в рабочей области была какая-либо схема, то будет выдан запрос на сохранение.
Open	Загружает в рабочую область ранее созданную схему
Add	В текущую рабочую область добавляется содержимое ранее созданной схемы
Save	Сохраняет рабочую область для последующей работы. Если была новая схема, то надо будет ввести ее имя
Save As	Сохранение рабочей области под другим именем
Page Setup	Устанавливает границы листа при выводе на принтер
Print	Распечатка схемы, изображенной на рабочей области
Printer & System Config	Вызов панели инструментов ОС Windows
Exit	Выход из программы

2.3.1.4. Меню Edit

Меню Edit позволяет редактировать рабочую область.

Undo	Отменяет последнее действие
Cut	Вырезает выделение в буфер обмена
Copy	Копирует выделение в буфер обмена
Paste	Копирует из буфера обмена в рабочую область
Clear	Очищает выделение
Rotate 180	Разворачивает выделенное на 180 градусов
Create Compound Block	Объединяет выделенное в один составной блок
Rename Block	Переименовывает составной блок
Add Input	Добавляет дополнительные входы
Add Output	Добавляет дополнительные выходы
Remove Input	Удаляет вход, добавленный Add Input
Remove	Удаляет выход, добавленный Add Output

Output	
Preference	Задает параметры рабочей области: полосы прокрутки, точность отображения чисел, радиус автосоединения, цветное отображение и т.д.
Repaint Screen	Обновление экрана
Delete One Block	Удаляет указанный блок и все его связи

2.3.1.5. Меню Simulate

Меню Simulate предназначено для установки метода интегрирования, запуска и остановки моделирования системы, представленной на рабочей области.

Go	Запуск моделирования в соответствии с установками: время моделирования, метод моделирования
Stop	Останавливает моделирование
Continue	Возобновляет моделирование после останова
Change parameters	Задает параметры моделирования
Control Panel	Выводит панель с кнопками Go, Snap State, Stop, Continue
Snap State	Запоминает начальные условия интеграторов

2.3.1.6. Меню Analyze

Меню Analyze предназначен для графического отображения функциональных процессов промоделированной системы. Для того чтобы задействовать меню Analyze, необходимо сначала промоделировать систему, т.е. воспользоваться функцией Simulate/Go.

Linearize	Проведение линеаризации произвольной системы, в том числе и нелинейной. Ответ выводится либо на экран, либо записывается в файл. Любую систему можно описать в виде: $$X' = AX + BU,$$ $$Y = CX + DU,$$ где X – вектор состояния системы; Y – вектор выходных координат системы; U – вектор входного воздействия; A, B, C, D – связующие матрицы
Select Input/Output Signal	Указывает входные и выходные части структурной схемы, которую необходимо линеаризовать
Transfer	Получение информации о передаточной функции системы или

Function Info	ее части. Необходимо, чтобы в выделенной части схемы присутствовал хотя бы один интегратор
Root Locus	Построение корневого годографа
Root Locus Response Gain	Задание границ коэффициента усиления и числа точек для расчета корневого годографа
Frequency Response	Построение ЛАЧХ, ЛФЧХ в заданном диапазоне
Frequency Range	Указание диапазона частот для построения ЛЧХ
Preference	Задание машинного нуля для собственных чисел, коэффициентов числителя и знаменателя передаточной функции

2.3.2. Задания для самостоятельной работы

2.3.2.1. Решение неоднородных дифференциальных уравнений

Система управления, как любая динамическая система, описывается неоднородными дифференциальными уравнениями [2.5]. Для линейных систем управления эти уравнения имеют постоянные коэффициенты

$$y^{(n)} + a_{n-1}y^{(n-1)} + \ldots + a_1 y' + a_0 y = b_m f^{(m)} + b_{m-1} f^{(m-1)} + \ldots + b_1 f' + b_0 f +$$
$$+ g_k u^{(k)} + g_{k-1} u^{(k-1)} + \ldots + g_1 u' + g_0 u,$$

где $y(t)$, $y'(t)$,…, $y^{(n)}(t)$ – неизвестная функция времени и ее производные; $f(t)$ и $u(t)$ – известные функции.

В этой связи промоделируем решение обыкновенного неоднородного дифференциального уравнения

$$\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 5t + 1,$$

где y – некоторая искомая функция времени на отрезке [0; T] при нулевых начальных условиях. Для достижения поставленной цели необходимо проделать следующие процедуры:

1) разрешить заданное уравнение относительно наивысшей из производных

$$\dddot{y} = -3\ddot{y} - \dot{y} - 3y + 5t + 1;$$

2) составить схему модели (см. рис. 2.20), реализующую это уравнение, для чего выбрать следующие блоки:

- три интегратора (INTEGRATOR), которые разместим последовательно друг за другом;

- четыре блока с постоянными коэффициентами передачи (GAIN), которые равны соответствующим коэффициентам уравнения;
- четыре блока-сумматора (SUMMINGJUNCTION);
- один блок задания постоянной величины (CONST);
- блок источника сигнала (RAMP);
- блок вывода графиков (PLOT) для наблюдения процессов.

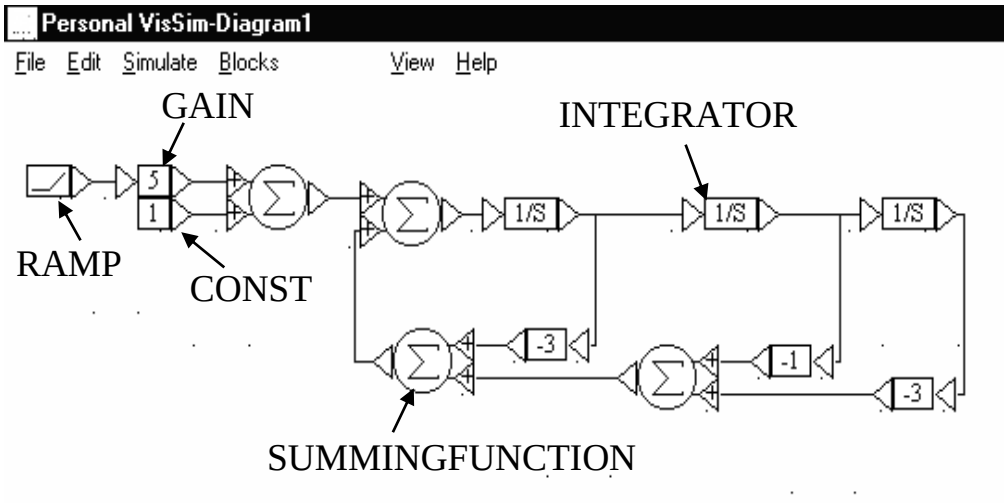


Рис. 2.20. Графическое представление уравнения

Входу крайнего левого интегратора соответствует переменная $\dddot{y}$, входу среднего интегратора – $\ddot{y}$, а входу крайнего правого интегратора – $\dot{y}$. Выход этого интегратора будет соответствовать переменной y;

3) соединить входы и выходы соответствующих блоков, для чего необходимо подвести курсор к выходу (входу) блока так, чтобы курсор принял вид черной вертикальной стрелки, нажать ЛК мыши и, удерживая ее, соединить выход (вход) одного блока с входом (выходом) другого.

С целью более упорядоченного расположения связей иной блок следует развернуть на 180 градусов, для чего необходимо выделить этот блок (либо с помощью клавиши Shift и щелчка ЛК мыши, либо обвести элемент прямоугольником, удерживая ЛК мыши) и затем нажать Alt+′стрелка влево′, или воспользоваться пунктом Rotate меню Edit. После выполнения всех соединений блоков получим следующее изображение на экране;

4) указать метод интегрирования и шаг разбиения по сетке времени, войдя в меню Simulate/Change parametrs (см. рис. 2.21). Например: диапазон моделирования (время начала и окончания от 0 до 50, шаг 0,05), алгоритм интегрирования (метод Рунге-Кутта) и т.д.;

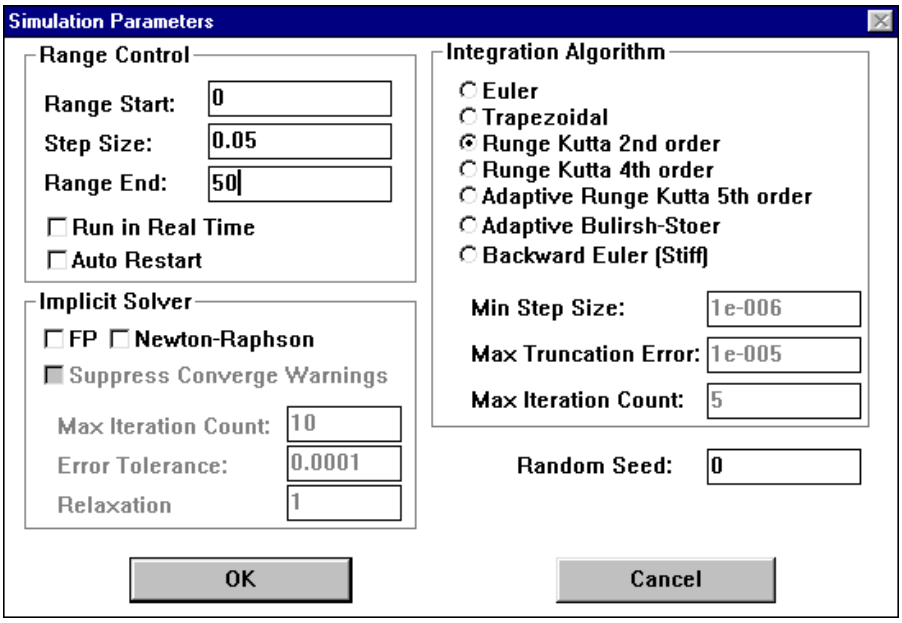


Рис. 2.21. Окно изменения параметров

5) инициировать процесс моделирования, войдя в меню Simulate/Go;

6) зафиксировать графический результат моделирования (Prt Scr) и проанализировать его с помощью блока Plot, входы которого могут быть соединены с любой точкой анализа. В данном примере для иллюстрации анализируется входной сигнал, заданный значением $5t + 1$, и выходной сигнал – y. Для иллюстрации модели, описанной неоднородным дифференциальным уравнением, необходимо сохранить схему и графический результат анализа данного уравнения, используя возможности Windows – Prt Scr.

2.3.2.2. Параметрический синтез устойчивой системы управления

Результаты, полученные в предыдущем разделе, свидетельствуют о неустойчивом характере решения дифференциального уравнения, так как найденная функция $y(t)$ изменяется по незатухающему гармоническому закону. Необходимо проанализировать влияние изменения коэффициентов этого уравнения на колебательные процессы. Из теории [2.5] следует, что для этого все коэффициенты однородного дифференциального уравнения

$$a_3\dddot{y} + a_2\ddot{y} + a_1\dot{y} + a_0 y = 0$$

должны быть больше единицы и выполнялось условие

$$a_2a_1 - a_3a_0 > 0\,.$$

Анализ влияния изменения коэффициентов уравнения a_0, a_1, a_2 на колебательные процессы необходимо провести по однофакторному эксперименту. Для этого необходимо выполнить следующие процедуры:

1) выбрать коэффициент a_0 для вариаций;

2) щелкнуть ПК мыши по соответствующему блоку GAIN и изменить его коэффициент на +20 %;

3) инициировать процесс моделирования, войдя в меню Simulate/Go;

4) для сохранения выходного графика при последующем изменении параметров схемы необходимо в блоке Plot/Plot Setup установить флаг для параметра Over Plot (рис. 2.22);

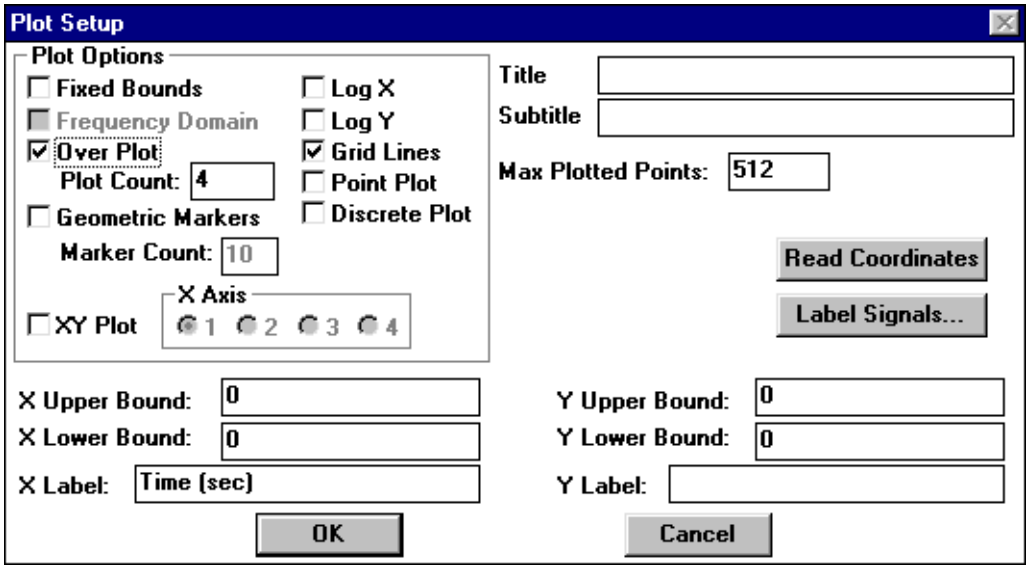


Рис. 2.22. Окно установки флага для параметра Over Plot

5) изменить коэффициент выбранного блока GAIN на –20 %;

6) инициировать процесс моделирования, войдя в меню Simulate/Go;

7) зафиксировать графический результат моделирования (Prt Scr), обратите внимание, что в блоке Plot должно быть два графика для +20 и –20 % значений коэффициента a_0;

8) проанализировать устойчивость системы для двух вариантов коэффициента a_0: +20 и –20 %;

9) задать начальное значение варьируемого коэффициента;

10) для того чтобы при последующем изменении параметров схемы в блоке Plot не сохранились старые графики, необходимо в блоке Plot/Plot Setup сбросить флаг для параметра Over Plot;

11) выбрать другой коэффициент для вариаций;

12) повторить пункты 2 – 10 для коэффициентов a_1 и a_2;

13) обобщить полученные результаты и сделать выводы об устойчивости системы при вариации параметров системы.

2.3.2.3. Моделирование систем и объектов управления, представленных передаточными функциями

В предыдущих пунктах рассмотрен один из способов решения дифференциальных уравнений. Второй способ моделирования заключается в решении уравнения, представленного передаточной функцией, которая

связывает изображение входной и выходной величин при нулевых начальных условиях. В данном случае исходное дифференциальное уравнение может иметь только две функции – искомую и известную

$$y^{(n)} + a_{n-1}y^{(n-1)} + \ldots + a_1 y' + a_0 y = b_m f^{(m)} + b_{m-1} f^{(m-1)} + \ldots + b_1 f' + b_0 f \ .$$

Формально получить передаточную функцию можно путем замены оператора дифференцирования d/dt на комплексную переменную s в степени, равной порядку производной (теоретическое обоснование такой операции вытекает из преобразований Лапласа) [2.6]. В соответствии с этим перепишем уравнение и определим передаточную функцию

$$s^n Y(s) + a_{n-1}s^{n-1}Y(s) + \ldots + a_1 sY(s) + a_0 Y(s) =$$
$$= b_m s^m F(s) + b_{m-1}s^{m-1}F(s) + \ldots + b_1 sF(s) + b_0 F(s).$$

Решим это уравнение относительно изображения $Y(s)$ и получим

$$Y(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} F(s),$$

где $W(s) = \dfrac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}$ – передаточная функция.

Изменим уравнение, с которым работали в пункте 2.3.2.1, исключив вторую известную функцию (единичный скачок) $\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 5t$. С использованием аппарата передаточных функций это уравнение можно привести к виду

$$X(s) = \frac{5}{s^3 + 3s^2 + s + 3} \times \frac{1}{s^2},$$

где $1/s^2$ – изображение временной функции *t*. Уравнению $\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 1$ соответствует уравнение

$$X(s) = \frac{1}{s^3 + 3s^2 + s + 3} \times \frac{1}{s}.$$

Таким образом, полное решение анализируемого уравнения

$$X(s) = \frac{5}{s^3 + 3s^2 + s + 3} \times \frac{1}{s^2} + \frac{1}{s^3 + 3s^2 + s + 3} \times \frac{1}{s}.$$

В процессе моделирования нет необходимости задавать изображения известных функций. Они в данном случае задаются двумя блоками: RAMP (5*t*) и STEP (1) как функции времени. Для моделирования необходимо проделать следующие процедуры:

1). Вставить два блока TRANSFERFUNCTION, описывающие передаточные функции, из меню INTEGRATION. Для ввода параметров передаточной функции необходимо нажать ПК мыши, появится диалоговое

окно (рис. 2.23). В поле GAIN первого блока TRANSFERFUNCTION необходимо ввести число 5, в поле Numertor необходимо ввести значение числителя 1, в поле Denominator – значение знаменателя в виде разделенных пробелами чисел при s от старшей степени к младшей: 1 3 1 3. В поле GAIN второго блока TRANSFERFUNCTION необходимо ввести число 1, в поле Numertor необходимо ввести значение числителя 1, в поле Denominator – значение знаменателя в виде разделенных пробелами чисел при s от старшей степени к младшей: 1 3 1 3.

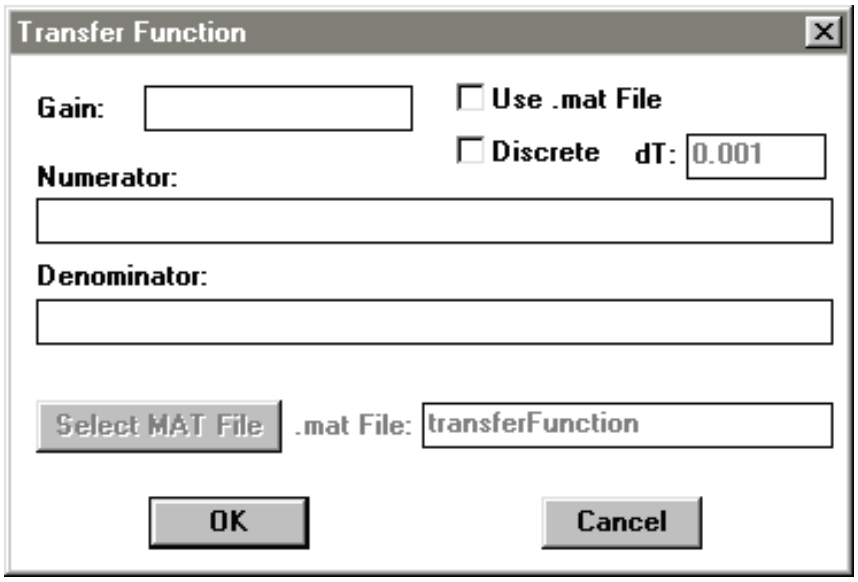


Рис. 2.23. Ввод передаточной функции

2). Вставить блок SUMMINGJUNCTION. Соединить его входы с выходами блоков TRANSFERFUNCTION.

3). Вставить блок RAMP. Его выход соединить со входом блока TRANSFERFUNCTION, который имеет коэффициент передачи, равный 5.

4). Вставить блок STEP. Его выход соединить со входом другого блока TRANSFERFUNCTION, у которого коэффициент передачи равен 1.

5). Вставить блок PLOT. На его вход подать сигнал с выхода сумматора.

6). Установить в меню Simulate/Change prameters (см. рис. 2.21):

- шаг по сетке времени – Step Size – 0,1;
- алгоритм интегрирования – классический Рунге-Кутта – Runge-Kutt 4^{th} order;
- начало временного отрезка – Range Start – 0;
- конец временного отрезка – Range End – 10;
- нажать клавишу OK.

7). Инициировать процесс моделирования, войдя в меню Simulate/Go. В элементе PLOT появится графическая интерпретация решения уравнения.

8). Решить это же уравнение методом Эйлера с тем же шагом интегрирования.

9). Установить в знаменателях передаточных функций коэффициенты, найденные в предыдущем разделе, которые обеспечивают устойчивость решения дифференциального уравнения. Инициировать процесс моделирования, войдя в меню Simulate/Go.

10). Исключить из схемы блок RAMP, чтобы получить переходную характеристику системы. Инициировать процесс моделирования, войдя в меню Simulate/Go.

11). Решить уравнение, полученное в п. 10, но при правой части, равной 10, на отрезке времени [0;25] методом Эйлера с шагом 0,1; 0,01; 0,001 и методом Рунге-Кутта с шагом 0,01.

12). Обобщить результаты моделирования.

2.4. МОДЕЛИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ С ПОМОЩЬЮ MATLAB

2.4.1. Описание пакета программ MATLAB

2.4.1.1. Рабочая область MATLAB

Пакет программ MATLAB предназначен для решения большого круга задач, в том числе для моделирования процессов в объектах управления, а также процессов в системах управления объектами, которые можно описать дифференциальными уравнениями. Работа пакета базируется на интегрировании дифференциальных уравнений, вводимых в пакет с помощью графических блоков структурных схем с соответствующими значениями параметров. Работа пакета программ MATLAB описана в [2.7]. Данный пакет предоставляет инженеру-разработчику или исследователю богатую библиотеку различного рода блоков структурных схем и вспомогательных блоков для наиболее полного анализа динамических систем.

После запуска MATLAB на экране появится окно, представленное на рис. 2.24.

Для создания новой модели необходимо в данном окне щелкнуть левой кнопкой мыши на иконке  (Simulink) и в появившемся окне, представленном на рис. 2.25, выбрать в меню File\New\Model (для быстрого создания можно использовать сочетание клавиш Ctrl+N). В результате вместе с окном Simulink Library Browser на экране отобразится окно, которое является рабочей областью создания модели (см. рис. 2.26).

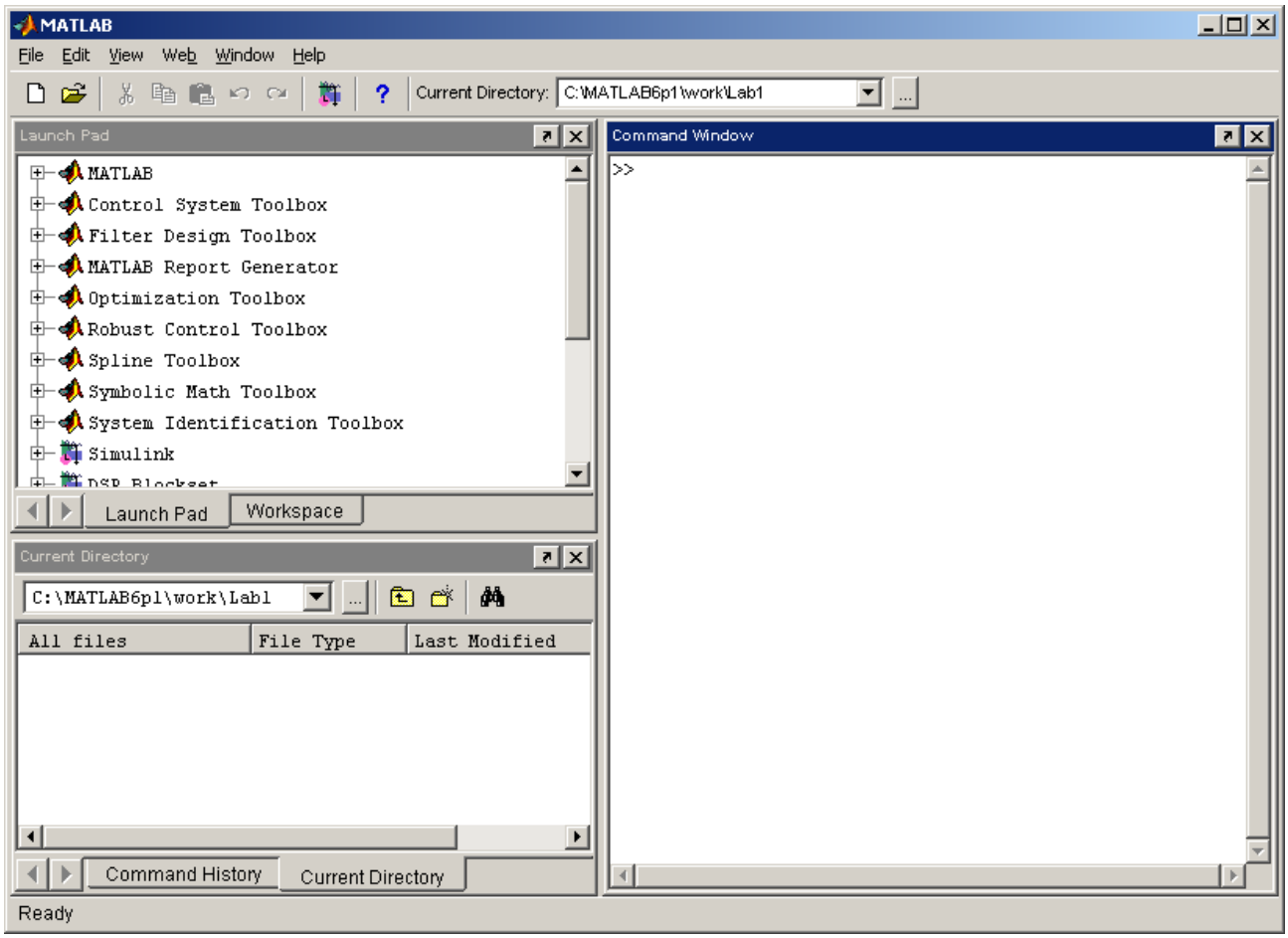


Рис. 2.24. Окно запуска MATLAB

Выбор составных блоков модели осуществляется с помощью браузера библиотеки Simulink (Simulink Library Browser) (см. рис. 2.25), состоящего из девяти разделов: Continuous, Discrete, Functions & Tables, Math, Nonlinear, Signals & Systems, Sinks, Sources, Subsystems.

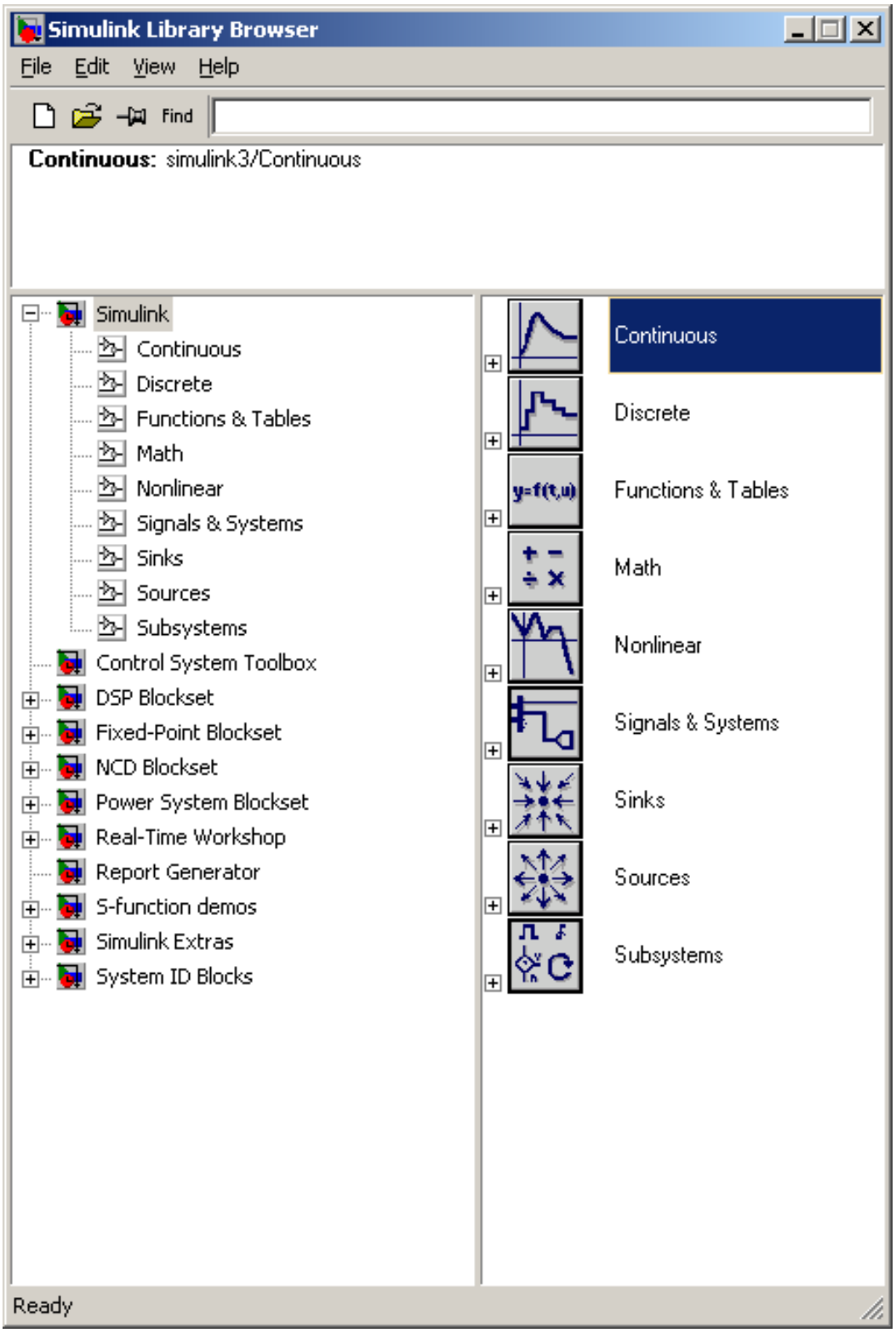


Рис. 2.25. Окно браузера библиотеки Simulink

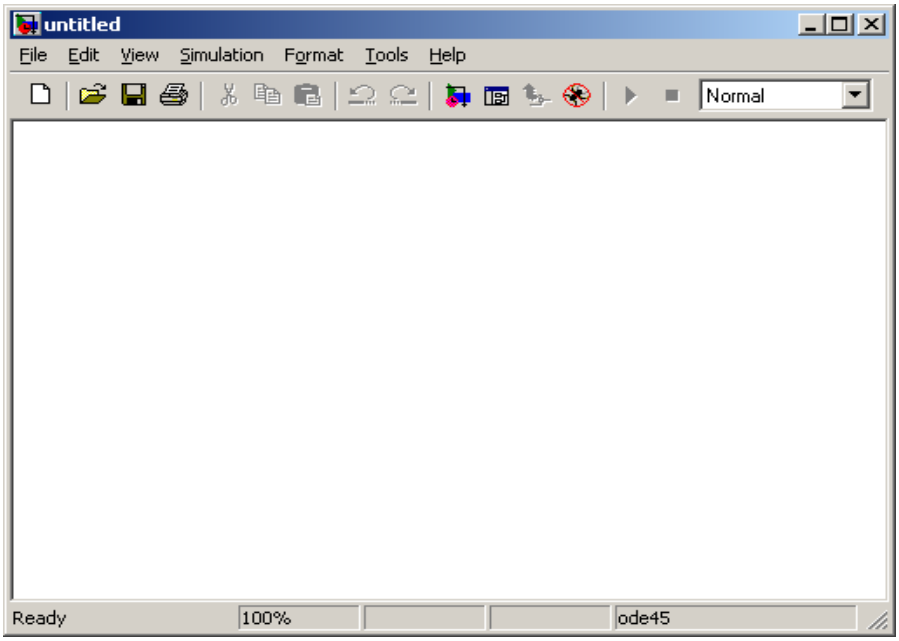


Рис. 2.26. Окно рабочей области MATLAB

2.4.1.2. Раздел Continuous

Раздел *Continuous* (непрерывные системы) содержит блоки, которые можно условно разделить на две группы: блоки, непосредственно предназначенные для описания непрерывных систем, и блоки общего назначения, которые могут быть использованы в модели любой системы.

1). *Derivative* (производная) – блок вычисления производной входного сигнала по времени.

2). *Integrator* (интегратор) – сумматор непрерывного действия.

3). *Memory* (регистр) – блок задержки входного сигнала на один шаг моделирования.

4). *State-Space* (пространство состояний) – блок описания системы в векторно-матричной форме.

5). *Transfer Fcn* (передаточная функция) – блок задания передаточных функций.

6). *Transport Delay* (транспортное запаздывание) – блок, реализующий произвольную задержку входного сигнала.

7). *Variable Transport Delay* (переменное транспортное запаздывание) – блок, реализующий динамически изменяемую задержку входного сигнала.

8). *Zero-Pole* (область перехода) – блок описания непрерывной системы в виде нулей и полюсов.

2.4.1.3. Раздел Discrete

В раздел *Discrete* (дискретные системы) входят блоки, с помощью которых в модели может быть описано поведение дискретных систем.

1). *Discrete Transfer Fcn* (дискретная передаточная функция) – блок задания дискретной передаточной функции.

2). *Discrete Zero-Pole* (дискретная область перехода) – блок описания дискретной системы в виде нулей и полюсов.

3). *Discrete Filter* (дискретный фильтр) – блок создания дискретного фильтра, порядок и свойства которого задаются полиномом от частного $1/z$ в числителе передаточной функции.

4). *Discrete State-Space* (дискретное пространство состояний) – блок формирования дискретного пространства состояний.

5). *Discrete-Time Integrator* (дискретный интегратор времени) – блок дискретного интегрирования времени. Обычно он служит для управления логической работой модели, например для остановки процесса моделирования по заданному значению интеграла времени.

6). *First-Order Hold* (экстраполятор первого порядка) – блок задержки выходного сигнала на заданный промежуток времени и задания линейного изменения выходного сигнала на каждом такте дискретизации, в соответствии с крутизной входного сигнала в данный момент времени.

7). *Unit Delay* (дискретная единичная задержка) – блок, обеспечивающий задержку входного сигнала на один шаг эталонного времени.

8). *Zero-Order Hold* (экстраполятор первого порядка) – блок задержки выходного сигнала на заданный промежуток времени. Данный блок оставляет неизменными значения выходного сигнала на каждом такте дискретизации.

2.4.1.4. Раздел Functions & Tables

Раздел *Functions & Tables* (функции и таблицы) содержит компоненты функций и таблиц. Блоки функций позволяют вводить в модели практически любые функции, что имеет большое значение в математическом моделировании различных объектов и систем. Блоки таблиц обеспечивают задание табличных данных с различной размерностью, а также позволяют использовать линейную или сплайновую интерполяцию и экстраполяцию данных.

1). *Direct Look-Up Table (n-D)* (многомерная таблица с прямым доступом) – блок, позволяющий задавать многомерные таблицы с прямым доступом к их элементам.

2). *Fcn* (функция) – блок задания функций одной переменной или ряда переменных $u(i)$.

3). *Interpolation (n-D) using PreLook-Up* (многомерная интерполяционная таблица) – блок, позволяющий создать многомерную интерполяционную таблицу для представления в табличном виде функций ряда переменных.

4). *Look-Up Table* (одномерная таблица) – блок задания в табличной форме некоторых данных, представляющих ряд значений функции одной переменной.

5). *Look-Up Table (2-D)* (двумерная таблица) – блок задания таблиц, представляющих значения функций двух переменных.

6). *Look-Up Table (n-D)* (многомерная таблица) – блок табличного представления данных, имеющий расширенные возможности в части задания размерности таблиц и интерполяции и экстраполяции их данных.

7). *MATLAB Fcn* (блок задания функции MATLAB) – блок задания одной переменной или ряда переменных $u(i)$ по правилам, принятым для языка программирования базовой системы MATLAB. Это означает, что в теле функции могут встречаться как встроенные функции системы MATLAB, так и любые процедуры и функции, реализованные в виде *m*-файлов.

8). *Polynomial* (полином) – блок задания степенного многочлена $P(u)$ в виде вектора его коэффициентов.

9). *PreLook-Up Index Search* (работа с индексами) – блок вычисления принадлежности одномерных данных к ближайшим узлам при приближении к ним снизу, а также контроль разности (дистанции) в относительных единицах между данными и значениями этих узловых точек.

10). *S-Function* (S-функция) – блок задания *S*-функций.

2.4.1.5. Раздел Math

Раздел *Math* (математические блоки) содержит блоки, которые реализуют элементарные (алгебраические и тригонометрические) функции, а также операции математической логики и могут быть использованы в модели любой системы.

1). *Abs* (абсолютное значение) – блок формирования абсолютного значения входного сигнала.

2). *Algebraic Constraint* (алгебраический корень) – блок, предназначенный для отыскания корней алгебраических уравнений.

3). *Bitwise Logical Operator* (побитовый логический оператор) – блок, содержащий набор основных логических операторов: AND, OR, NAND, NOR, XOR, NOT. Этот блок выполняет вышеперечисленные операции побитово.

4). *Combinatorial Logic* (комбинаторная логика) – блок, обеспечивающий преобразование входного сигнала в соответствии с правилами, задаваемыми так называемой таблицей истинности.

5). *Complex to Magnitude-Angle* (выделение из комплексного числа амплитуды и фазы) – блок, позволяющий получить из входной комплексной величины амплитудно-фазовые характеристики сигнала.

6). *Complex to Real-Imag* (выделение из комплексного числа действительной и мнимой части) – блок, служащий для выделения из комплексного числа действительной и мнимой части.

7). *Dot Product* (свертка) – блок, обеспечивающий скалярное умножение (свертку) двух векторов, подаваемых на его входы.

8). *Gain* (умножитель) – блок, выполняющий роль умножителя сигнала, поступающего на его вход.

9). *Logical Operator* (логический оператор) – блок, содержащий набор основных логических операторов: AND, OR, NAND, NOR, XOR, NOT.

10). *Magnitude-Angle to Complex* (преобразование амплитуды и фазы в комплексное число) – блок, позволяющий получить из амплитудно-фазовой характеристики сигнала комплексную величину.

11). *Math Function* (математическая функция) – блок, позволяющий использовать для преобразования входного сигнала элементарные нетригонометрические функции (вычисление экспоненты, натурального и десятичного логарифмов, возведение в степень, извлечение квадратного корня и т.д.).

12). *Matrix Gain* (матричный множитель) – блок, предназначенный для умножения входного сигнала на матрицу (или вектор).

13). *MinMax* (минимум или максимум) – блок, обеспечивающий поиск минимального или максимального элемента входного вектора.

14). *Product* (умножение) – блок, позволяющий выполнять умножение или деление нескольких входных сигналов (величин).

15). *Real-Imag to Complex* (преобразование действительной и мнимой части в комплексное число) – блок, позволяющий получить из действительной и мнимой части комплексное число.

16). *Relational Operator* (оператор отношения) – блок, реализующий операции отношения (>, <, ?, == (тождественно), != (не равно) над двумя входными сигналами.

17). *Rounding Function (функция округления)* – блок, содержащий различные функции округления значения амплитуды выходного сигнала.

18). *Sign* (знак) – блок, выполняющий проверку знака выходного сигнала.

19). *Slider Gain* (настраиваемый умножитель) – блок, позволяющий динамически изменять коэффициент умножения в процессе моделирования.

20). *Sum* (сложение) – блок, выполняющий суммирование входных сигналов.

21). *Trigonometric Function* (тригонометрическая функция) – блок, обеспечивающий преобразование входного сигнала с помощью одной из тригонометрических функций.

2.4.1.6. Раздел Nonlinear

Раздел *Nonlinear* (нелинейные системы) содержит блоки, которые используются для описания нелинейных систем.

1). *Backlash* (люфт) – блок, имитирующий эффект возникновения люфта, при котором создается передаточная характеристика гистерезисного типа.

2). *Coulomb & Viscous Friction* (блок фрикционных эффектов) – блок, служащий для моделирования фрикционных эффектов.

3). *Dead Zone* (зона нечувствительности) – блок, заменяющий значение входного сигнала, лежащее в заданном диапазоне, нулем.

4). *Manual Switch* (ручной переключатель) – блок, позволяющий вручную выбирать один из двух входных портов, сигнал с которого будет передаваться на выход блока.

5). *Multiport Switch* (многоходовой переключатель) – блок, позволяющий задавать требуемое количество входов (портов) и подключать тот или иной вход к выходу в зависимости от уровня сигнала на управляющем входе.

6). *Quantizer* (квантователь) – блок, служащий для квантования меняющихся сигналов с одинаковым шагом по уровню.

7). *Rate Limiter* (ограничитель скорости) – блок, предназначенный для контроля и ограничения скорости изменения (то есть первой производной) входного сигнала.

8). *Relay* (реле) – релейный блок, имеющий разрывную передаточную функцию с гистерезисом (или без него).

9). *Saturation* (насыщение) – блок, представляющий собой нелинейное устройство, сигнал на выходе которого равен входному сигналу до тех пор, пока не достигнет порогов ограничения: верхнего Upper limit или нижнего Lower limit.

10). *Switch* (ключ) – блок управляемого переключателя с тремя входами: двумя (крайними) для сигналов данных и одним (средним) для сигналов управления.

2.4.1.7. Раздел Signals & Systems

Раздел *Signals & Systems* (сигналы и системы) – один из самых больших разделов, содержит блоки, которые могут быть использованы для описания широкого класса систем: непрерывных, дискретных и др.

1). *Bus Selector* (селектор шины) – блок, выделяющий из группы сигналов один заданный сигнал.

2). *Data Store Memory* (память для данных) – блок, обеспечивающий хранение данных на интервале моделирования.

3). *Data Store Read* (чтение данных) – блок, обеспечивающий чтение данных на интервале моделирования.

4). *Data Store Write* (запись данных) – блок, обеспечивающий запись данных на интервале моделирования. Блоки 2, 3, 4 используются совместно.

5). *Data Type Conversion* (приведение типа данных) – блок, обеспечивающий приведение типа сигнала к требуемому.

6). *Demux* (разделитель) – блок, разделяющий векторный входной сигнал на несколько сигналов.

7). *Function-Call Generator* (генератор вызова функций) – блок, запускающий подключенные к нему подсистемы с заданной периодичностью.

8). *From* (принять) – блок, предназначенный для обмена (принятия) данными между различными компонентами *S*-модели с учетом доступности (видимости) этих данных.*Goto* (передать) – блок, предназначенный для обмена (передачи) данными между различными компонентами *S*-модели с учетом доступности (видимости) этих данных.

9). *Goto Tag Visibility* (видимость блока передачи) – блок, предназначенный для обмена данными между различными компонентами S-модели с учетом доступности (видимости) этих данных. Блоки 8, 9, 10 используются совместно.

10). *Hit Crossing* (индикатор пересечения) – блок, позволяющий идентифицировать момент времени, когда входной сигнал «пересекает» некоторое значение.

11). *IC* (начальное состояние) – блок, позволяющий устанавливать произвольное начальное состояние входного сигнала.

12). *Matrix Concatenation* (объединение сигналов в матрицу) – блок объединения сигналов в матрицу образует из ряда векторов матрицу, обеспечивая их объединение (конкатенацию) по горизонтали и по вертикали.

13). *Merge* (поглотитель) – блок, отбирающий из входных сигналов один, значение которого было вычислено последним.

14). *Model Info* (информация о модели) – блок, позволяющий контролировать информацию, относящуюся к разработке модели (версия, имя разработчика и т.п.).

15). *Mux* (смеситель) – блок, объединяющий входные сигналы в один векторный сигнал.

16). *Probe* (проверка) – блок, предназначенный для просмотра основных характеристик сигнала (длительность, тип), а также величины шага моделирования.

17). *Reshape* (восстановление) – блок, позволяющий восстановить входной сигнал.

18). *Selector* (селектор) – блок, выделяющий из входного вектора заданные элементы.

19). *Signal Specification* (спецификация сигналов) – блок, служащий для распознавания сигнала на его входе и для выдачи спецификации.

20). *Width* (длительность) – блок вычисления длительности сигнала, поступающего на его вход; вычисленное значение выводится непосредственно внутри блока.

2.4.1.8. Раздел Sinks

В разделе *Sinks* (получатели) размещены блоки-приемники сигналов.

1). *Display* (экран) – блок, предназначенный для отображения численных значений величин.

2). *Scope* (осциллограф) – блок, позволяющий в процессе моделирования наблюдать динамику интересующих исследователя характеристик системы.

3). *Stop Simulation* (остановка моделирования) – блок, позволяющий прервать моделирование при выполнении тех или иных условий.

4). *Terminator* (ограничитель) – блок, использующийся в качестве «заглушки» для выходных портов.

5). *To File* (запись в файл) – блок, обеспечивающий сохранение промежуточных и/или выходных результатов моделирования в файле.

6). *To Workspace* (запись в рабочую область) – блок, обеспечивающий сохранение промежуточных и/или выходных результатов моделирования в рабочей области MATLAB.

7). *XY Graph* (двумерный график) – блок, обеспечивающий создание двумерных графиков в прямоугольной системе координат.

2.4.1.9. Раздел Sources

Блоки, входящие в раздел *Sources* (источники), предназначены для формирования сигналов, которые обеспечивают управление работой модели в целом или отдельных ее частей.

1). *Band-Limited White Noise* («белый шум» с ограниченной полосой) – блок-генератор «белого шума» с ограниченной полосой.

2). *Chirp Signal* (гармонический сигнал) – блок-источник гармонических колебаний переменной частоты.

3). *Clock* (часы) – блок-генератор непрерывного временного сигнала.

4). *Constant* (константа) – блок, обеспечивающий формирование постоянного скалярного или векторного сигнала.

5). *Digital Clock* (цифровые часы) – блок-источник дискретного временного сигнала.

6). *From Workspace* (ввод из рабочей области) – блок, реализующий ввод в модель данных непосредственно из рабочей области MATLAB.

7). *From File* (ввод из файла) – блок, реализующий ввод в модель данных, хранящихся в MAT-файле.

8). *Ground* (земля) – блок, использующийся в качестве «заглушки» для входных портов.

9). *Pulse Generator* (импульсный генератор) – блок-генератор импульсных сигналов.

10). *Ramp* (возбудитель) – блок-генератор линейно возрастающего (убывающего) сигнала вида $F(t)=k \cdot t$.

11). *Random Number* (случайное число) – блок-источник случайного сигнала, служащий для создания случайного сигнала с равномерным распределением уровня.

12). *Repeating Sequence* (периодический сигнал) – блок-генератор периодического дискретного сигнала произвольной формы.

13). *Signal Generator* (генератор сигнала) – блок-генератор непрерывного сигнала произвольной формы.

14). *Sine Wave* (генератор гармонических колебаний) – блок-генератор гармонических колебаний.

15). *Step* («ступенька») – блок-источник воздействия в виде одиночного перепада.

16). *Uniform Random Number* (равномерное случайное число) – блок-источник дискретного сигнала, амплитуда которого является равномерно распределенной случайной величиной.

2.4.1.10. Раздел Subsystems

Раздел *Subsystems* (подсистемы) содержит блоки, предназначенные для создания подсистем (субмоделей). Внутри подсистем первого уровня могут располагаться подсистемы второго уровня и т.д. Сложная система набирается из отдельных систем – модулей, каждый из которых, в свою очередь, является системой или устройством.

2.4.2. Задания для самостоятельной работы

2.4.2.1. Решение неоднородных дифференциальных уравнений

Система управления, как и любая динамическая система (см. п.2.3.2.1), описывается неоднородными дифференциальными уравнениями [2.5]:

$$y^{(n)} + a_{n-1}y^{(n-1)} + \ldots + a_1 y' + a_0 y = b_m f^{(m)} + b_{m-1} f^{(m-1)} + \ldots + b_1 f' + b_0 f +$$
$$+ g_k u^{(k)} + g_{k-1} u^{(k-1)} + \ldots + g_1 u' + g_0 u,$$

где $y(t)$, $y'(t)$, …, $y^{(n)}(t)$ – неизвестная функция времени и ее производные; $f(t)$ и $u(t)$ – известные функции.

Для линейных систем управления эти уравнения имеют постоянные коэффициенты: $a_{n\text{-}1}$, …, a_1, a_0, b_m, $b_{m\text{-}1}$, …, b_1, b_0, g_k, $g_{k\text{-}1}$, …, g_1, g_0.

Промоделируем динамическую систему, описанную обыкновенным неоднородным дифференциальным уравнением

$$\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 5t + 1,$$

где y – некоторая искомая функция времени на отрезке $[0;T]$ при нулевых начальных условиях.

Для достижения поставленной цели необходимо проделать следующие процедуры:

1) разрешить заданное уравнение относительно наивысшей из производных:

$$\dddot{y} = -3\ddot{y} - \dot{y} - 3y + 5t + 1;$$

2) составить схему модели (см. рис. 2.27), реализующую это уравнение, для чего собрать следующие блоки:

- три интегратора (*Integrator*), которые разместим последовательно друг за другом;
- четыре блока с постоянными коэффициентами передачи (*Gain*), которые равны соответствующим коэффициентам уравнения;
- четыре блока-сумматора (*Sum*);
- один блок задания постоянной величины (*Constant*);
- блок линейно нарастающего сигнала (*Ramp*);
- блок вывода графиков (*Scope*) для наблюдения процессов.

Входу крайнего левого интегратора соответствует переменная $\dddot{y}$, входу среднего интегратора – $\ddot{y}$, а входу крайнего правого интегратора – $\dot{y}$. Выход этого интегратора будет соответствовать переменной y;

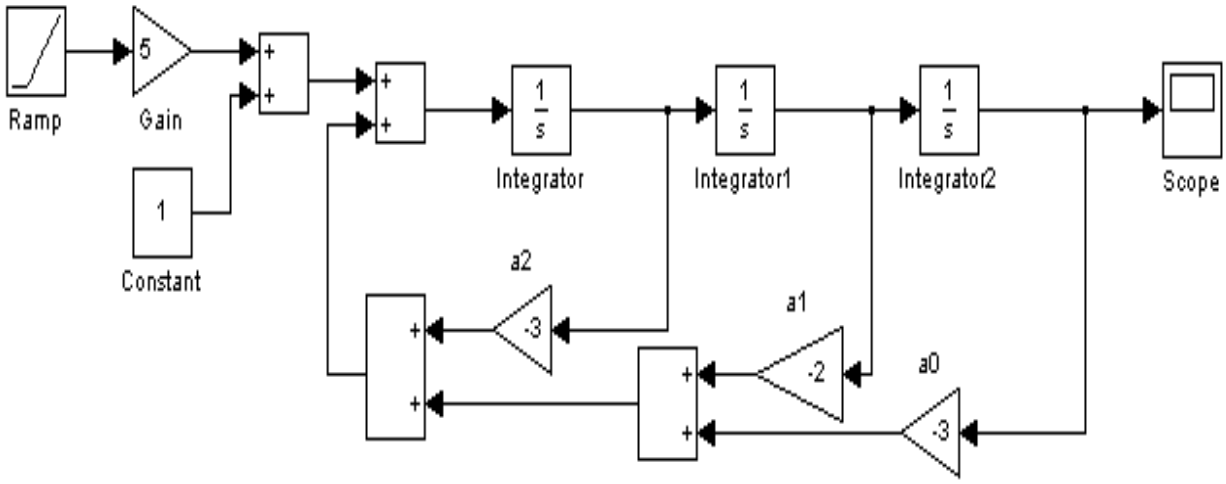


Рис. 2.27. Графическое представление уравнения

3) соединить входы и выходы соответствующих блоков, для чего необходимо подвести указатель мыши к выходу (входу) блока так, чтобы указатель принял вид перекрестия, нажать левую кнопку мыши и, удерживая ее, соединить выход (вход) одного блока с входом (выходом) другого. С целью более упорядоченного расположения связей любой блок следует развернуть на 180 °, для чего необходимо выделить этот блок (либо с помощью щелчка левой кнопкой мыши, либо обвести элемент прямоугольником, удерживая левую кнопку мыши) и затем нажать *Ctrl+R*. После выполнения всех соединений блоков получим следующее изображение на экране;

4) указать диапазон моделирования (время начала и окончания, шаг), алгоритм интегрирования и шаг разбиения по сетке времени, войдя в меню *Simulation\Simulation parameters* (см. рис. 2.28). В окне *Simulation Parameters* можно задать следующие параметры моделирования:

- *Start time* – значение времени, начиная с которого производится моделирование;
- *Stop time* – значение времени, при котором моделирование останавливается;
- *Type* – задается тип шага моделирования: *Veriable-step* – моделирование с переменным шагом, *Fixed-step* – моделирование с фиксированным шагом;
- Раскрывающийся список справа от *Type* – позволяет задать алгоритм интегрирования;
- *Max step size* – максимальный шаг интегрирования;
- *Min step size* – минимальный шаг интегрирования;
- *Initial step size* – начальный шаг интегрирования;
- *Relative tolerance* – относительная погрешность интегрирования;
- *Absolute tolerance* – абсолютная погрешность интегрирования;

- *Refine output* (скорректированный вывод) – позволяет изменять дискретность регистрации модельного времени и параметров модели;
- *Refine factor* (корректирующий множитель) – по умолчанию это значение равно 1; указывает на то, что регистрация производится для каждого очередного значения модельного времени;

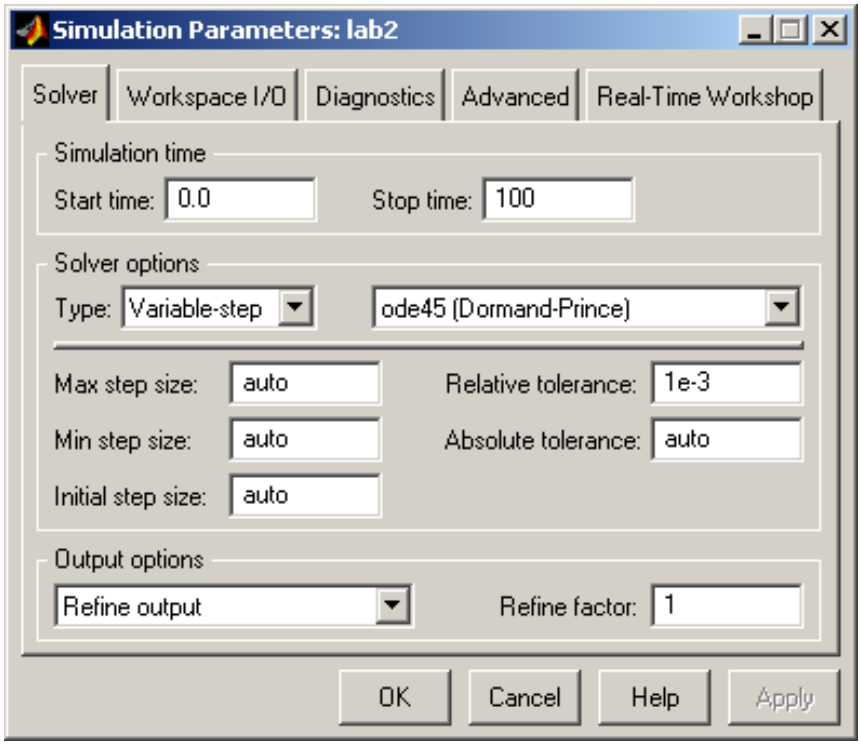


Рис. 2.28. Окно изменения параметров

5) инициировать процесс моделирования, войдя в меню *Simulation\Start* либо нажав *Ctrl+T*;
6) зафиксировать графический результат моделирования и проанализировать его с помощью блока *Scope*, входы которого могут быть соединены с любой точкой анализа.

В данной самостоятельной работе анализируется входной сигнал, заданный значением $5t + 1$ и выходной сигнал y.

2.4.2.2. Параметрический синтез устойчивости системы управления

Результаты, полученные в предыдущем разделе, свидетельствуют о неустойчивом характере динамической системы, описанной дифференциальным уравнением, так как найденная функция $y(t)$ изменяется по незатухающему гармоническому закону. Необходимо изменить коэффициенты этого уравнения так, чтобы колебательные процессы затухали. Для этого все коэффициенты однородного дифференциального уравнения

$$a_3\dddot{y} + a_2\ddot{y} + a_1\dot{y} + a_0 y = 0$$

должны быть положительны и выполнялось условие [2.5]

$$a_2 a_1 - a_3 a_0 > 0.$$

Для решения поставленной задачи необходимо выполнить следующие процедуры:

1) проделать задания пунктов 1 – 4 предыдущего раздела;
2) выбрать коэффициент для вариаций;
3) двойным щелчком левой кнопки мыши по соответствующему блоку *Gain* изменить его коэффициент;
4) инициировать процесс моделирования, войдя в меню Simulation\Start;
5) зафиксировать графический результат моделирования, проанализировать устойчивость системы;
6) повторить пункты 3 – 5 настоящего раздела, изменяя коэффициент выбранного блока *Gain* на ±20 %;
7) задать начальное значение варьируемого коэффициента;
8) выбрать другой коэффициент для вариаций;
9) повторить пункты 3 – 7;
10) обобщить полученные результаты.

Таким образом, для нахождения устойчивого решения схемы необходимо провести однофакторный эксперимент со следующим планом:

1) $a_0 = 3{,}6$; $a_1 = 1$; $a_2 = 3$; $a_3 = 1$;
2) $a_0 = 2{,}4$; $a_1 = 1$; $a_2 = 3$; $a_3 = 1$;
3) $a_0 = 3$; $a_1 = 1{,}2$; $a_2 = 3$; $a_3 = 1$;
4) $a_0 = 3$; $a_1 = 0{,}8$; $a_2 = 3$; $a_3 = 1$;
5) $a_0 = 3$; $a_1 = 1$; $a_2 = 3{,}6$; $a_3 = 1$;
6) $a_0 = 3$; $a_1 = 1$; $a_2 = 2{,}4$; $a_3 = 1$.

2.4.2.3. Моделирование систем и объектов управления, представленных передаточными функциями

В предыдущих пунктах рассмотрен один из способов моделирования динамической системы, описанной с помощью дифференциальных уравнений. Второй способ моделирования динамической системы заключается в решении уравнения, представленного передаточной функцией, которая связывает изображение входной и выходной величин при нулевых начальных условиях (см. п. 2.3.2.3). В данном случае исходное дифференциальное уравнение может иметь только две функции – искомую $y(t)$ и известную $f(t)$:

$$y^{(n)} + a_{n-1}y^{(n-1)} + \ldots + a_1 y' + a_0 y = b_m f^{(m)} + b_{m-1} f^{(m-1)} + \ldots + b_1 f' + b_0 f\ .$$

Формально получить передаточную функцию можно путем замены оператора дифференцирования *d*/*dt* на комплексную переменную *s* в степени, равной порядку производной (теоретическое обоснование такой операции вытекает из преобразований Лапласа) [2.6]. В соответствии с этим перепишем уравнение и определим передаточную функцию:

$$s^n Y(s) + a_{n-1}s^{n-1}Y(s) + \ldots + a_1 sY(s) + a_0 Y(s) =$$
$$= b_m s^m F(s) + b_{m-1}s^{m-1}F(s) + \ldots + b_1 sF(s) + b_0 F(s).$$

Решим это уравнение относительно изображения $Y(s)$ и получим

$$Y(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} F(s),$$

где $W(s) = \dfrac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}$ – передаточная функция.

Изменим уравнение, с которым работали в пункте 2.4.2.1, исключив вторую известную функцию (единичный скачок) $\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 5t$. С использованием аппарата передаточных функций это уравнение можно привести к виду

$$X(s) = \frac{5}{s^3 + 3s^2 + s + 3} \times \frac{1}{s^2},$$

где $1/s^2$ – изображение временной функции *t*.

Уравнение $\dddot{y} + 3\ddot{y} + \dot{y} + 3y = 1$ соответствует уравнению

$$X(s) = \frac{1}{s^3 + 3s^2 + s + 3} \times \frac{1}{s}.$$

Таким образом, полное решение анализируемого уравнения имеет следующий вид:

$$X(s) = \frac{5}{s^3 + 3s^2 + s + 3} \times \frac{1}{s^2} + \frac{1}{s^3 + 3s^2 + s + 3} \times \frac{1}{s}.$$

Изображение временной функции $1/s^2$ задается блоком *Ramp*, изображение постоянной константы – блоком *Step*.

Для моделирования необходимо проделать следующие процедуры:

1) вставить два блока *Transfer Fcn* из библиотеки *Simulink Continuous*. Для ввода передаточной функции необходимо дважды щелкнуть левой кнопкой мыши на только что вставленном блоке. В поле *Numerator* первого блока необходимо ввести число 5, в поле *Denominator* – значение знаменателя в виде разделенных пробелами чисел при *s* от старшей степени к младшей: 1 3 1 3. В поле *Numerator* второго блока необходимо ввести число 1, в поле *Denominator* – значение знаменателя в виде разделенных пробелами чисел при *s* от старшей степени к младшей: 1 3 1 3;
2) вставить блок *Sum* из библиотеки *Math*. Соединить его с выходами передаточных функций;
3) вставить блок *Ramp* из библиотеки *Sources*. Его выход соединить с входом блока *Transfer Fcn*, который имеет коэффициент передачи, равный 5;
4) вставить блок *Step* из библиотеки *Sources*. Его выход соединить с входом другого блока *Transfer Fcn*, у которого коэффициент передачи равен 1;
5) вставить блок *Scope* из библиотеки *Sinks*. На его вход подать сигнал с выхода сумматора. В результате схема модели примет вид, представленный на рис. 2.29;

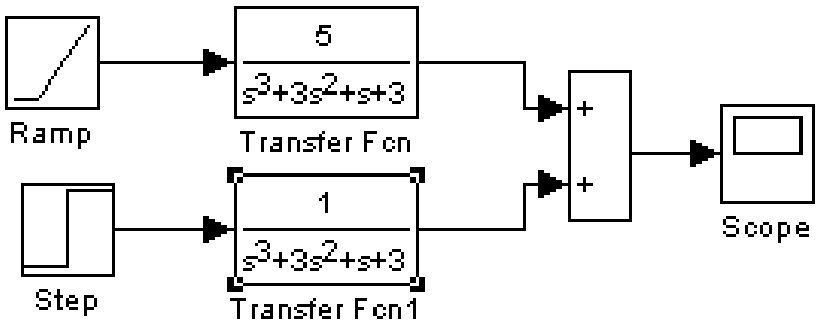


Рис. 2.29. Модель эксперимента

6) установить в меню *Simulation\Simulation parameters* (см. рис. 2.28):

- в параметрах расчета (*solver options*) тип расчета (*Type*) – фиксированный шаг (*Fixed-step*);
- шаг по сетке (*Fixed step size*) – 0,1;
- алгоритм интегрирования (*ode*4 (*Runge-Kutta*) – классический Рунге-Кутта;
- начало временного отрезка (*Start Time*) – 0;
- конец временного отрезка (*Stop Time*) – 50;

7) инициировать процесс моделирования, войдя в меню *Simulation\Start* либо нажав *Ctrl+T*. В элементе *Scope* появится графическая интерпретация решения уравнения;

8) решить это уравнение методом Эйлера (*ode*1 (*Euler*) с тем же шагом интегрирования;

9) установить в знаменателях передаточных функций найденные в предыдущем разделе коэффициенты, которые обеспечивают устойчивость решения дифференциального уравнения. Инициировать процесс моделирования, войдя в меню *Simulation\Start*;

10) исключить из схемы блок *Ramp*, чтобы получить переходную характеристику системы. Инициировать процесс моделирования, войдя в меню *Simulation\Start*;

11) решить это же уравнение, но при правой части, равной 10, на отрезке времени [0;50] методом Эйлера (*ode*1 (*Euler*) с шагом 0,1; 0,01; 0,001 и методом Рунге-Кутта (*ode*4 (*Runge-Kutta*) с шагом 0,01;

12) обобщить результаты моделирования.

2.5. ДЕТЕРМИНИРОВАННОЕ МОДЕЛИРОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ

2.5.1. Методика моделирования систем автоматического регулирования

Любую модель строят в зависимости от цели [2.4]. В качестве цели моделирования ставится задача изучения какой-либо стороны функционирования объекта (контроль параметров, оценка характеристик, управление объектом, прогнозирование поведения объекта). Цель может быть сформулирована качественно или количественно. При количественной формулировке цели строят целевую функцию, которая точно отображает наиболее существенные факторы, влияющие на достижение цели. В качестве цели моделирования выберем изучение функционирования системы, а именно оценивание ее характеристик.

Далее определяются с целью функционирования системы, которая обеспечивала бы эффективную работу системы, например для системы автоматического регулирования (САР) – устойчивость системы, высокая чувствительность к изменению параметров, оптимальные характеристики качества системы.

С учетом имеющихся ресурсов в качестве метода решения задачи выберем метод имитационного моделирования, который позволяет решать задачи анализа систем, включая задачи оценки: вариантов структуры системы, эффективности различных алгоритмов управления системой, влияния изменения различных параметров системы. Имитационное моделирование может быть положено также в основу структурного, алгоритмического и

параметрического синтеза систем, когда требуется создать систему с заданными характеристиками при определенных ограничениях, которая является оптимальной по некоторым критериям оценки эффективности.

Основные этапы имитационного моделирования системы:

- построение концептуальной модели системы и ее формализация;
- алгоритмизация модели системы и ее машинная реализация;
- получение и интерпретация результатов моделирования системы.

Для реализации имитационной модели используются средства вычислительной техники, которые могут лишь помочь с точки зрения эффективности реализовать сложную модель, но не позволяют подтвердить правильность модели. Только на основе отработанных данных, опыта исследователя можно с достоверностью оценить адекватность модели по отношению к реальному процессу.

После построения модели требуется провести эксперимент с моделью. Если цель эксперимента – изучение влияния переменной x на переменную y, то x – фактор, а y – реакция. Каждый фактор x_i, $i = \overline{1,k}$, может принимать в эксперименте одно из нескольких значений, называемых уровнями. Фиксированный набор уровней факторов определяет одно из возможных состояний рассматриваемой системы.

Связь между уровнями факторов и реакцией (откликом) системы можно представить в виде соотношения

$$y_l = \Psi_l(x_1, x_2, \ldots x_k), \quad l = \overline{1,m}. \tag{2.1}$$

Функцию Ψ_l, связывающую реакцию с факторами, называют функцией реакции. Исследователю заранее неизвестен вид зависимостей Ψ_l, поэтому используют приближенные соотношения:

$$\tilde{y}_l = \varphi_l(x_1, x_2, \ldots x_k), \quad l = \overline{1,m}. \tag{2.2}$$

Зависимости φ_l находятся по данным эксперимента.

При построении плана эксперимента необходимо помнить, что целями проведения машинных экспериментов с моделью системы являются либо получение зависимости реакции от факторов для выявления особенностей изучаемого процесса функционирования системы (анализ модели), либо нахождение такой комбинации значений факторов, которая обеспечивает экстремальное значение реакции (синтез модели).

При проведении машинного эксперимента с моделью для оценки некоторых характеристик процесса функционирования исследуемой системы необходимо выявить влияние факторов, находящихся в функциональной связи с искомой характеристикой. Для этого необходимо: отобрать факторы x_i, $i = \overline{1,k}$, влияющие на искомую характеристику, и описать функциональную

зависимость; установить диапазон изменения факторов $x_{i\min} \pm x_{i\max}$; определить координаты точек факторного пространства $\{x_1, x_2, \ldots x_k\}$, в которых следует проводить эксперимент; оценить необходимое число реализации и их порядок в эксперименте.

Эксперимент, в котором реализуются все возможные сочетания уровней факторов, называется полным факторным экспериментом (ПФЭ). Если выбранная модель планирования линейная, то для оценки коэффициентов модели используется план эксперимента с варьированием всех k факторов на двух уровнях, т.е. $q = 2$. Такие планы называются планами типа 2^k, где $N = 2^k$ – число всех возможных испытаний. Для получения коэффициентов линейной модели варьирование факторов осуществляется одновременно на двух уровнях: нижнем $x_{i\text{н}}$ и верхнем $x_{i\text{в}}$ – симметрично расположенных относительно точки центра эксперимента x_{i0}, $i = \overline{1,k}$.

Однофакторный (классический) эксперимент предназначен для получения линейной экспериментальной факторной модели и предусматривает поочередное варьирование каждого из факторов при фиксированных на некотором уровне значениях остальных факторов. Фактор x_i варьируют на двух уровнях: нижнем $x_{i\text{н}}$ и верхнем $x_{i\text{в}}$, а все остальные при этом должны находиться в точке центра эксперимента x_{i0}, $j \neq i$.

При синтезе системы на основе проведения машинных экспериментов с моделью возникает задача анализа чувствительности модели к вариациям ее параметров. Под анализом чувствительности машинной модели понимают проверку устойчивости результатов моделирования, т.е. характеристик процесса функционирования системы, полученных при проведении имитационного эксперимента, по отношению к возможным отклонениям параметров машинной модели $\Delta\vec{h} = (\Delta h_1, \ldots, \Delta h_n)$ от истинных их значений $\vec{h} = (h_1, \ldots, h_n)$.

Малым отклонениям $\Delta\vec{h}$ будут соответствовать изменения характеристик $\vec{q}(\vec{h})$, которые в практических расчетах можно оценить величиной

$$\Delta\vec{q} = \vec{q}'(\vec{h})\Delta\vec{h} + r_0, \qquad (2.3)$$

где $\vec{q}'(\vec{h}) = (\partial q(\vec{h})/\partial h_1, \ldots, \partial q(\vec{h})/\partial h_n)$; r_0 – остаточный член второго порядка малости относительно вариации, который используется для проверки точности решения.

Частная производная $\vec{q}'(\vec{h})$ определяется в точках, соответствующих номинальным значениям параметров $\vec{h}_{\text{ном}}$. Если $\vec{h}_{\text{ном}} = \vec{h}^*$, где $\vec{h}^*$ – оптимальные параметры системы по показателю $\vec{q}(\vec{h})$, то $\vec{q}'(\vec{h}_{\text{ном}}) = 0$ и необходимо проводить оценку с использованием второй производной $\vec{q}''(\vec{h}_{\text{ном}})$. Таким образом, частные производные $\vec{q}'(\vec{h})$, $\vec{q}''(\vec{h})$

количественно характеризуют чувствительность машинной модели к изменениям ее параметров.

Большие отклонения характеристик $\vec{q}(\vec{h})$ при малых вариациях $\Delta\vec{h}$ свидетельствуют о неустойчивости модели по отношению к этим вариациям. Для получения оценок $\tilde{\vec{q}}(\vec{h})$ показателя $\vec{q}(\vec{h})$ удобно рассматривать зависимые реализации внешних воздействий при различных $\vec{h}$ и проводить соответствующую обработку результатов машинного эксперимента с моделью.

Вопросы качества САР рассмотрены подробно в п. 2.5.2.4.

2.5.2. Модели систем автоматического регулирования

2.5.2.1. Элементы одноконтурной САР

Системы автоматического регулирования (САР) являются разновидностью систем управления. Основная функция САР состоит в поддержании выходной величины системы вблизи заданного значения. Одноконтурная САР является простейшей из всех прочих САР. По этой причине элементы моделирования систем автоматического управления целесообразно изучать на базе такой САР.

Структурная схема простейшей одноконтурной САР представлена на рис. 2.30, где введены следующие обозначения: g – задающее воздействие (уставка); $f_{вн}$ – внутреннее возмущение по каналу регулирования; ε – сигнал ошибки; U – управляющее воздействие на объект управления (ОУ); y – выходной сигнал ОУ; $R(s)$ – передаточная функция (ПФ) регулятора; $W_{об}(s)$ – ПФ объекта управления по управляющему воздействию.

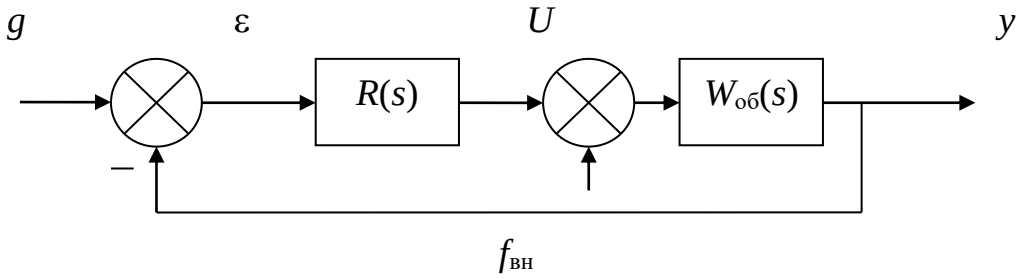


Рис. 2.30. Структурная схема одноконтурной САР

Для составления модели одноконтурной САР проводится синтез, при котором выбирают закон регулирования и определяют настройки блока регулирования. Во время анализа САР определяют ее устойчивость и качество по переходному процессу. Процедуры анализа являются основными элементами процесса моделирования САР. Для синтеза и анализа САР с одноконтурными регуляторами разработан программный модуль *SIANRG*. В этом модуле реализованы различные формы математического представления ОУ и блоков регулирования, которые далее именуются моделями соответствующих элементов САР [2.8].

2.5.2.2. Модели объекта управления

ОУ описывается передаточными функциями, которые характеризуют реакцию выхода системы на управляющее и возмущающее воздействия. ОУ классифицируют по числу входов, выходов и связей между входами и выходами.

Различают следующие структуры ОУ:

- с одним входом и одним выходом (рис. 2.31, а);
- с двумя входами и одним выходом (рис. 2.31, б);
- с одним входом, двумя выходами и параллельным соединением звеньев (рис. 2.31, в);
- с одним входом, двумя выходами и последовательным соединением звеньев (рис. 2.31, г);
- с двумя входами и двумя выходами (рис. 2.31, д).

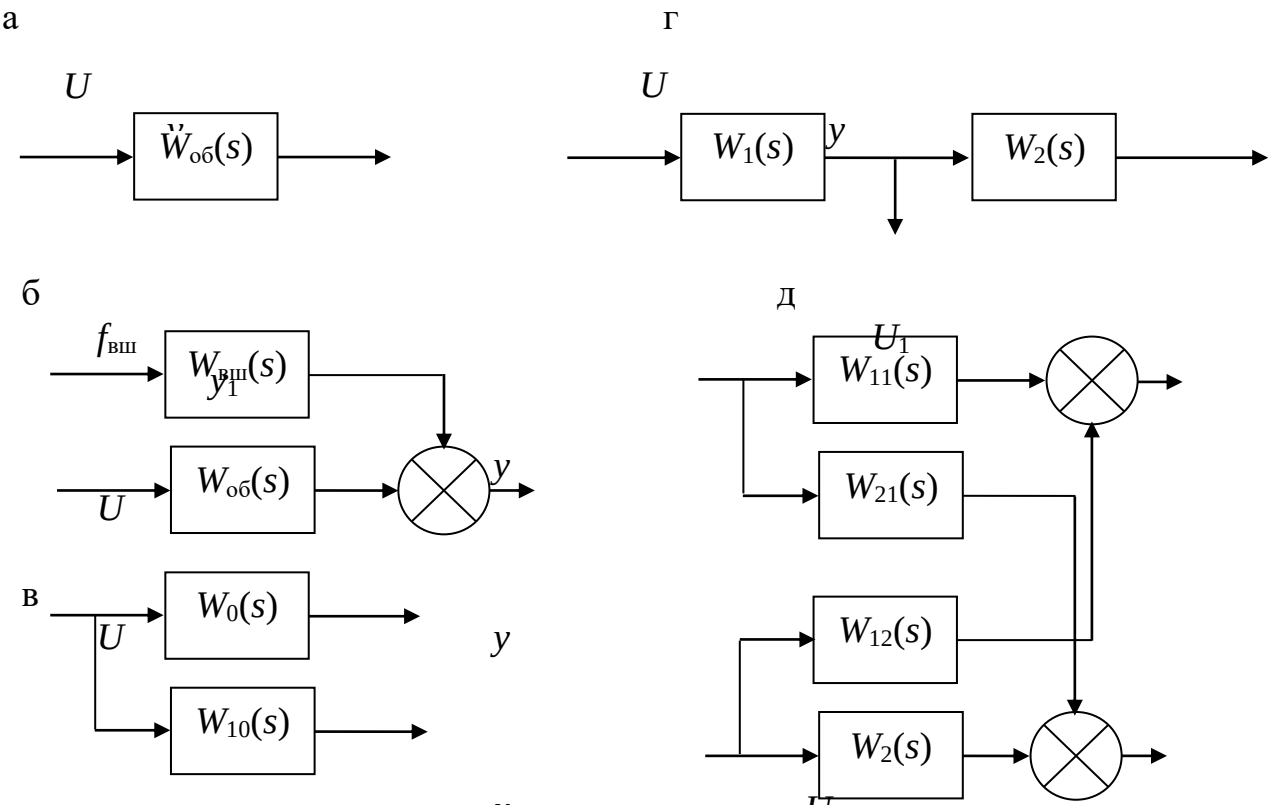


Рис. 2.31. Структурные схемы объектов управления

На рисунках переменной U обозначено управляющее воздействие на ОУ, которое реализуется с помощью регулирующего органа.

В общем случае между любой парой вход-выход может существовать динамическая связь (канал), которой на структурной схеме САР ставится в соответствие динамическое звено. ПФ такого звена в общем случае имеет вид:

$$W_{об}(s) = k_{об} \frac{A(s)}{B(s)} e^{-s\tau_{об}}, \qquad (2.4)$$

где $A(s)$ – полином числителя; $B(s)$ – полином знаменателя; $s = \sigma + j\omega$ – комплексная переменная преобразования Лапласа; $k_{об}$ – коэффициент передачи ОУ; $\tau_{об}$ – транспортное запаздывание ОУ, с. Как следует из рис. 2.31, ПФ канала ОУ может иметь другие индексы, помимо «об», которые использованы в формуле (2.4).

Полиномы числителя и знаменателя могут быть представлены в одной из трех форм:

$$A(s) = a_0 + a_1 s + \ldots + a_{m-1} s^{m-1} + a_m s^m; \qquad (2.5)$$

$$B(s) = b_0 + b_1 s + \ldots + b_{n-1} s^{n-1} + b_n s^n, \qquad (2.5')$$

или

$$A(s) = (s - \alpha_1)(s - \alpha_2)\ldots(s - \alpha_m); \qquad (2.6)$$

$$B(s) = (s - \beta_1)(s - \beta_2)\ldots(s - \beta_n), \qquad (2.6')$$

или

$$A(s) = (T_1^a s + 1)(T_2^a s + 1)\ldots(T_m^a s + 1); \qquad (2.7)$$

$$B(s) = (T_1^b s + 1)(T_2^b s + 1)\ldots(T_n^b s + 1), \qquad (2.7')$$

где a_i, b_j – коэффициенты полиномов; α_i, β_j – корни этих полиномов (в общем случае – комплексные); T_i^a, T_j^b – постоянные времени, с; m, n – степени полиномов.

В программном модуле предусмотрена возможность ввода полиномов числителя и знаменателя в трех перечисленных формах. При переходе от одной формы к другой следует помнить о необходимости корректировки значения коэффициента передачи объекта управления.

Особенности полинома знаменателя ПФ ОУ позволяют разделить ОУ на два класса:

1) статические ($b_0 \neq 0$);

2) астатические ($b_0 = 0$).

Статические ОУ называются также ОУ с самовыравниванием.

Зачастую $W_{об}(s)$ записывают таким образом, чтобы выполнялись соотношения $a_m = 1$; $b_n = 1$ или $a_0 = 1$; $b_0 = 1$. Для достижения таких соотношений необходимо вынести за скобки значения соответствующих коэффициентов и пересчитать значение $k_{об}$.

В некоторых методах расчета САР используют упрощенную форму ПФ для статического ОУ:

$$W_{об}(s) = k_{об} \frac{1}{T_{об}s + 1} e^{-s\tau_{об}} \quad (2.8)$$

или для астатического ОУ

$$W_{об}(s) = k_{об} \frac{1}{s} e^{-s\tau_{об}}. \quad (2.9)$$

Коэффициент передачи статического ОУ определяется по формуле

$$k_{об} = \frac{\Delta y}{\Delta U}$$

в рабочей точке регулировочной характеристики ОУ $y(U)$ и измеряется в единицах выходной величины на процент хода регулирующего органа (РО). Приращения Δy и ΔU выбирают вблизи их номинальных значений.

2.5.2.3. Модели регуляторов

Регулятор содержит следующие функциональные элементы САР: суммирующий элемент, блок регулирования, исполнительный механизм, задатчик (если уставки регулятору задает оператор вручную).

Регулирующие блоки САР обеспечивают типовые законы регулирования:

1) пропорциональный, или П-регулятор;

2) интегральный, или И-регулятор;

3) пропорционально-интегральный, или ПИ-регулятор;

4) пропорционально-интегрально-дифференциальный или ПИД-регулятор;

5) пропорционально-дифференциальный, или ПД-регулятор.

Передаточные функции динамических звеньев, соответствующих регулирующим блокам, определяются следующим отношением:

$$R(s) = \frac{U(s)}{\varepsilon(s)},$$

где $\varepsilon(s)$ – преобразование Лапласа входного сигнала блока регулирования; $U(s)$ – преобразование Лапласа выходного сигнала этого блока. ПФ регуляторов задают с использованием двух типов параметров, что приводит к двум формам их представления: форма 1 и форма 2.

ПФ регуляторов в форме 1 имеют следующий вид:

для П-регулятора $R(s) = k_р$; (2.10)

для ПИ-регулятора $R(s) = k_р \left(1 + \frac{1}{T_И s}\right)$; (2.11)

для ПИД-регулятора $R(s) = k_{\text{р}}\left(1+\frac{1}{T_{\text{И}}s}+T_{\text{Д}}s\right)$; (2.12)

или $R(s)=\frac{k_{\text{р}}T_{\text{И}}T_{\text{Д}}s^2+k_{\text{р}}T_{\text{И}}s+k_{\text{р}}}{T_{\text{И}}s}$; (2.12′)

для И-регулятора $R(s) = k_{\text{р1}}\frac{1}{s}$; (2.13)

для ПД-регулятора $R(s) = k_{\text{р}}(1+T_{\text{Д}}s)$. (2.14)

В этих формулах использованы следующие параметры: $k_{\text{р}}$ – коэффициент передачи регулятора в процентах хода РО на единицу регулируемой величины (так как выходная величина регулятора – ход исполнительного механизма – обычно равна входной величине ОУ – ходу регулирующего органа); $T_{\text{И}}$ – постоянная интегрирования (время изодрома), с; $T_{\text{Д}}$ – постоянная дифференцирования (время предварения), с. Для И-регулятора $T_{\text{И}} = 1$, а коэффициент передачи $k_{\text{р1}}$ измеряется в процентах хода РО на единицу регулируемой величины за секунду. В серийно выпускаемых регулирующих блоках перечисленные параметры передаточных функций необходимо настраивать в зависимости от динамических свойств ОУ. По этой причине такие параметры называются настройками регуляторов.

Сигнал на выходе ПИД-регулятора описывается формулой

$$U(t)=k_{\text{р}}\varepsilon(t)+\frac{k_{\text{р}}}{T_{\text{И}}}\int_0^t\varepsilon(\eta)d\eta+k_{\text{р}}T_{\text{Д}}\frac{d\varepsilon(t)}{dt}.$$

ПФ регуляторов в форме 2 (в *C*-параметрах) имеют следующий вид:

для П-регулятора $R(s) = C_1$; (2.15)

для ПИ-регулятора $R(s) = C_1 + \frac{C_0}{s}$; (2.16)

для ПИД-регулятора $R(s) = C_1 + \frac{C_0}{s} + C_2 s$, (2.17)

или $R(s)=\frac{C_2s^2+C_1s+C_0}{s}$. (2.17′)

В этих формулах использованы следующие параметры: C_1 – коэффициент пропорциональной составляющей; C_0 – коэффициент интегральной составляющей; C_2 – коэффициент дифференциальной составляющей. Перечисленные коэффициенты являются настройками регулятора. Для *C*-параметров сигнал на выходе ПИД-регулятора связан с сигналом ошибки следующим выражением:

$$U(t) = C_1\varepsilon(t) + C_0\int_0^t \varepsilon(\eta)d\eta + C_2\frac{d\varepsilon(t)}{dt}.$$

Настройки разных типов взаимосвязаны формулами:

$$C_1 = k_\text{р}; \qquad C_0 = \frac{k_\text{р}}{T_\text{И}}; \qquad C_2 = k_\text{р}T_\text{Д}$$

$$T_\text{И} = \frac{C_1}{C_0}; \qquad T_Д = \frac{C_2}{C_1} \tag{2.18}$$

В программном модуле ПФ ПД- и ПИД-регуляторов рассчитываются с физически реализуемыми дифференциальными составляющими

$$\frac{T_\text{Д}s}{0{,}25T_\text{Д}s+1} = \frac{C_1C_2s}{0{,}25C_2s+C_1} \tag{2.19}$$

вместо физически нереализуемых составляющих $T_\text{Д}s$ и C_2s.

Правая часть равенства (2.19) выведена на основании соотношения (2.18) для $T_\text{Д}$. С учетом указанных составляющих ПФ ПИД-регулятора имеет вид:

$$R(s) = \frac{1{,}25k_\text{р}T_\text{И}T_\text{Д}s^2 + k_\text{р}\left(0{,}25T_\text{Д} + T_\text{И}\right)s + k_\text{р}}{0{,}25T_\text{И}T_\text{Д}s^2 + T_\text{И}s} \tag{2.20}$$

или

$$R(s) = \frac{1{,}25C_1C_2s^2 + \left(0{,}25C_0C_2 + C_1^2\right)s + C_0C_1}{0{,}25C_2s^2 + C_1s}. \tag{2.21}$$

Для ПД-регулятора с учетом (2.19) можно найти следующие ПФ:

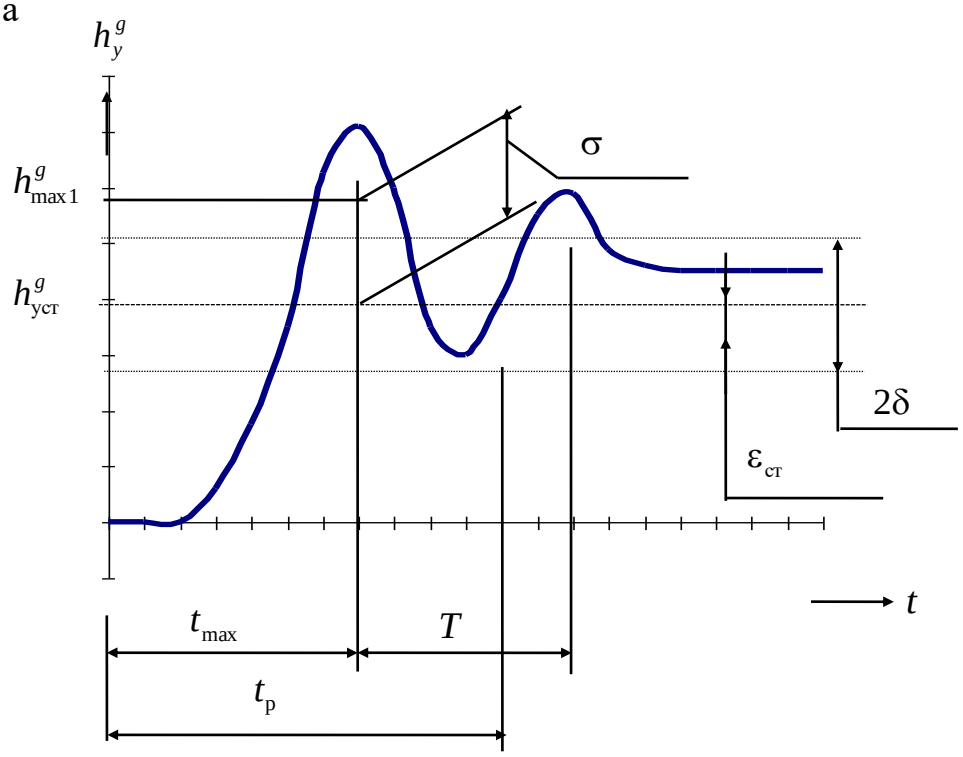
$$R(s) = \frac{1{,}25k_\text{р}T_\text{Д}s^2 + k_\text{р}}{0{,}25T_\text{Д}s+1} \quad \text{или} \quad R(s) = \frac{1{,}25C_1C_2s + C_1^2}{0{,}25C_2s + C_1}.$$

Формулы (2.20), (2.21) были использованы в программном модуле, формирующем коэффициенты передаточных функций регулирующих блоков по введенным настройкам.

2.5.2.4. Показатели качества САР

Целью моделирования САР является создание системы, удовлетворяющей определенным требованиям, которые формулируются в виде качественных показателей. Качество функционирования САР определяется по переходной характеристике (ПХ) $h(t)$ – реакции системы на типовое воздействие в виде единичной ступенчатой функции $1(t)$ – или некоторыми параметрами ПФ замкнутой системы. Ниже перечислены показатели качества, применяемые при моделировании и проектировании САР. Обозначения, используемые в формулах, пояснены на рис. 2.32 и 2.33.

а



б

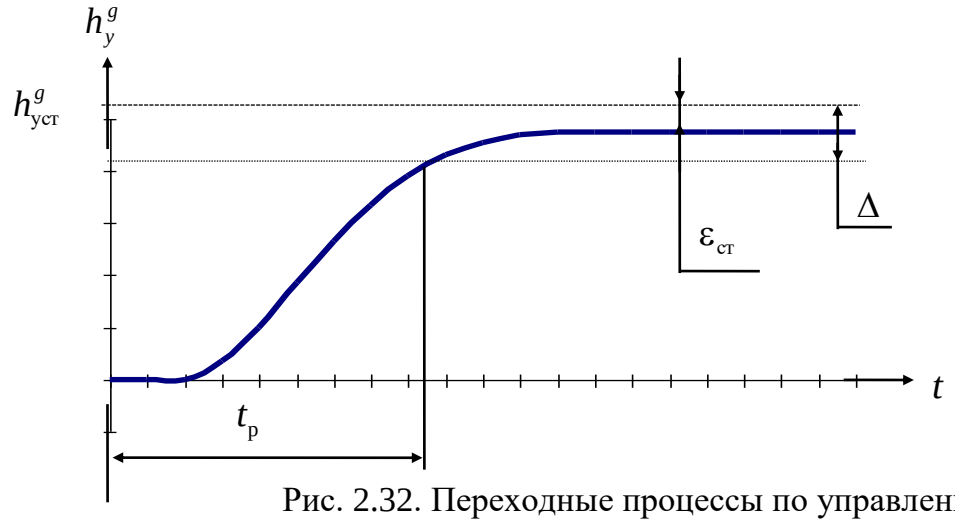


Рис. 2.32. Переходные процессы по управлению:
а – колебательный процесс; б – апериодический процесс

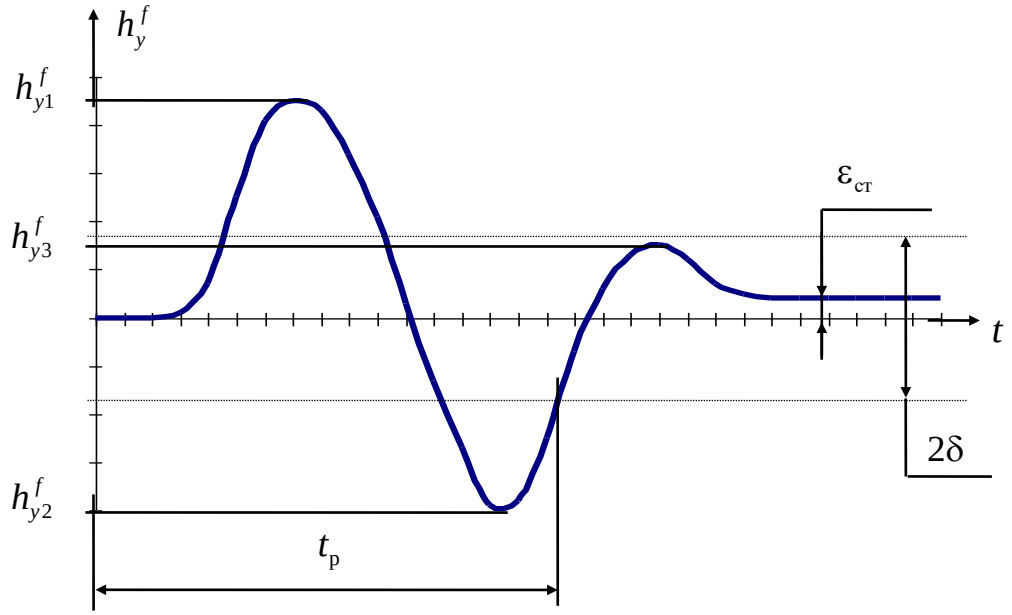
Максимальное динамическое отклонение по управлению в единицах регулируемой величины

$$\sigma = h_{max1}^g - h_{уст}^g . \qquad (2.22)$$

Максимальное динамическое отклонение по возмущению в единицах регулируемой величины

$$\sigma = h_{y1}^f . \qquad (2.23)$$

а



б

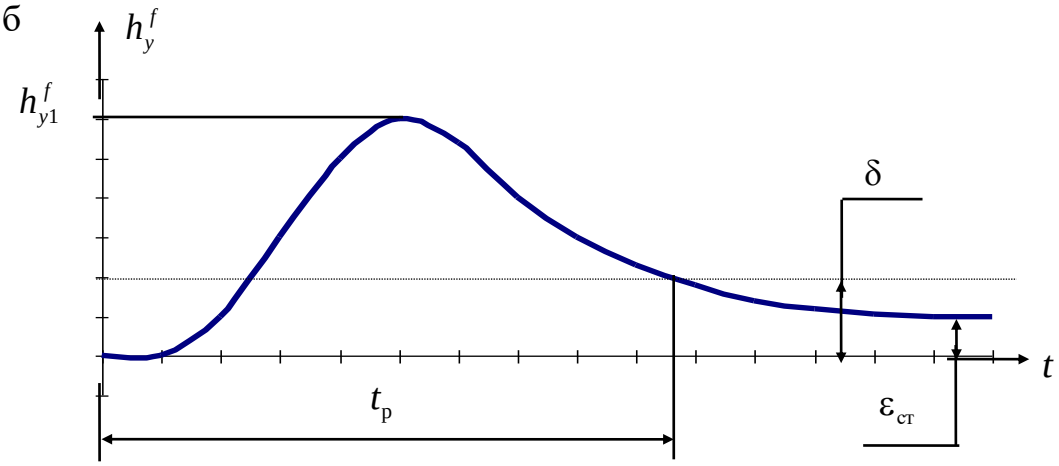


Рис. 2.33. Переходные процессы по возмущению:
а – колебательный процесс; б – апериодический процесс

Время регулирования $t_р$ в секундах, определяется при условии, когда сигнал зашел и больше не выходит из δ-трубки, относительно установившегося значения, где δ = 5 %.

Перерегулирование в процентах по управлению

$$\eta = 100\left(h_{\max 1}^g - h_{уст}^g\right)\Big/ h_{уст}^g . \qquad (2.24)$$

Перерегулирование в процентах по возмущению

$$\eta = 100\, h_{y2}^f \Big/ h_{y1}^f . \qquad (2.25)$$

Статическая ошибка $\varepsilon_{ст}$ в единицах регулируемой величины.

Частота колебаний $\omega = 2\pi/T$.

Время достижения первого максимума t_{max}.

Степень колебательности

$$m = \frac{\alpha}{\omega},$$

где α и ω – вещественная и мнимая части корней характеристического уравнения замкнутой САР, наиболее близко расположенных к мнимой оси.

Показатель колебательности M, определяемый как

$$M = \frac{A_{max}(\omega_p)}{A(0)},$$

где $A_{max}(\omega_p)$ – максимум АЧХ (амплитудно-частотной характеристики) замкнутой САР на резонансной ω_p или рабочей частоте; $A(0)$ – значение этой характеристики при нулевой частоте.

Интегральный квадратичный критерий качества

$$I = \int_0^\infty \varepsilon^2(t)dt, \qquad (2.26)$$

где $\varepsilon(t)$ – динамическая составляющая ошибки (без статической ошибки $\varepsilon_{ст}$).

Запасы устойчивости по амплитуде и фазе ΔA_0, $\Delta\varphi$, которые определяются по АФЧХ разомкнутой САР, не нашли применения при проектировании САР в промышленности.

Степень затухания переходного процесса по возмущению

$$\psi = \frac{h_{y1}^f - h_{y3}^f}{h_{y1}^f}. \qquad (2.27)$$

Первые шесть оценок относятся к прямым, так как они определяются непосредственно по ПХ. Остальные показатели являются косвенными оценками качества.

В ряде случаев качество САР задается типовым переходным процессом. Рассматривают три типовые ПХ:

1) апериодическая;
2) с 20%-ным перерегулированием;
3) с минимумом интегрального квадратичного критерия

$$\min I = \min \int_0^\infty \varepsilon^2(t)dt. \qquad (2.28)$$

2.5.2.5. Синтез регуляторов методом Циглера-Никольса

Синтез регулятора относится к этапу составления модели САР. Метод Циглера-Никольса, или метод незатухающих колебаний, применим для расчета САР с ОУ, передаточная функция которого задана полиномами произвольных степеней. В этом методе синтеза передаточную функцию блока регулирования следует представить в форме 2, или в *C*-параметрах.

Известно из критерия Найквиста, что если АФЧХ разомкнутой САР проходит через точку (-1, *j*0) комплексной плоскости, то система находится на границе устойчивости. В таком случае при $R(s) = C_1^{\text{кр}}$ можно записать равенство

$$W_{\text{рс}}(j\omega) = W_{\text{об}}(j\omega)C_1^{\text{кр}} = W(j\omega) = -1,$$

которое в показательной форме имеет вид:

$$A_{\text{об}}(\omega_{\text{кр}})e^{j\varphi(\omega_{\text{кр}})}C_1^{\text{кр}} = -1, \qquad (2.29)$$

где $A_{\text{об}}(\omega_{\text{кр}})$ – безразмерная амплитудная частотная характеристика объекта управления; $\varphi(\omega_{\text{кр}})$ – фазовая частотная характеристика ОУ.

Последнее равенство распадается на два уравнения:

$$\varphi(\omega_{\text{кр}}) = \pi \qquad \text{и} \qquad A_{\text{об}}(\omega_{\text{кр}})C_1^{\text{кр}} = 1, \qquad (2.30)$$

откуда следует

$$C_1^{кр} = \frac{1}{A_{\text{об}}(\omega_{\text{кр}})}. \qquad (2.31)$$

Этот метод применим только к тем САР, в которых имеется звено чистого запаздывания, или для систем, у которых АФЧХ разомкнутой САР пересекает отрицательную вещественную полуось.

В методе Циглера-Никольса расчет настроек регуляторов проводят в несколько этапов.

Этап 1. Определение критической частоты $\omega_{\text{кр}}$ из (2.30), на которой фазовый угол АФЧХ объекта управления равен 180 ° (см. рис. 2.34).

Условие $\varphi(\omega) = \pi$ выполняется на многих частотах, так как звено чистого запаздывания обусловливает спиралевидный характер АФЧХ при возрастании частоты. Критической частотой $\omega_{\text{кр}}$ будет наименьшая из этих частот. Остальные частоты будут ложными $\omega_{\text{л}}$ (см. рис. 2.34). Данное свойство необходимо учитывать при выборе $\omega_{\text{кр}}$.

В программном модуле автоматизированного моделирования САР частота $\omega_{\text{кр}}$ определяется в режиме интерактивного взаимодействия пользователя с программой. При этом пользователь выбирает нижнюю $\omega_{\text{н}}$ и верхнюю $\omega_{\text{в}}$ частоты (см. рис. 2.34).

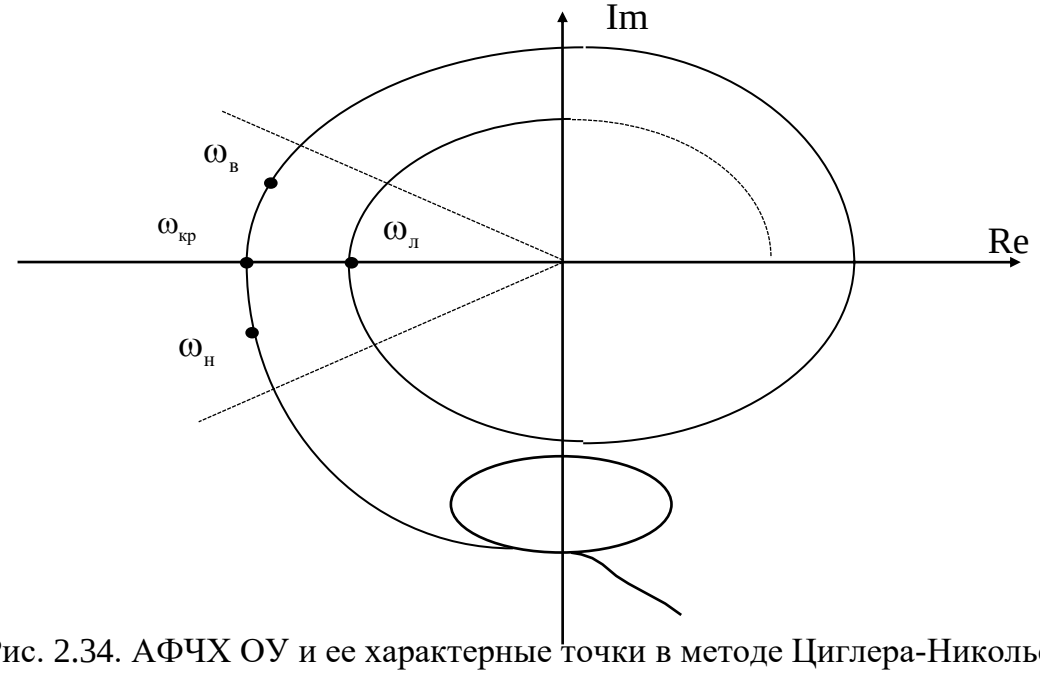


Рис. 2.34. АФЧХ ОУ и ее характерные точки в методе Циглера-Никольса

ПК для этих частот определяет фазовую частотную характеристику (ФЧХ) $\varphi_{рс}(\omega)$ разомкнутой САР с П-регулятором. Если ФЧХ удовлетворяет неравенствам

$$-180° < \varphi_{рс}(\omega_н) < -155°, \quad (2.32)$$

$$155° < \varphi_{рс}(\omega_в) < 180°, \quad (2.33)$$

то пользователь сообщает об этом факте программе, которая после принятия такого сообщения методом половинного деления вычисляет критическую частоту $\omega_{кр}$. Для предварительной оценки критической частоты можно воспользоваться графиком (рис. 2.35).

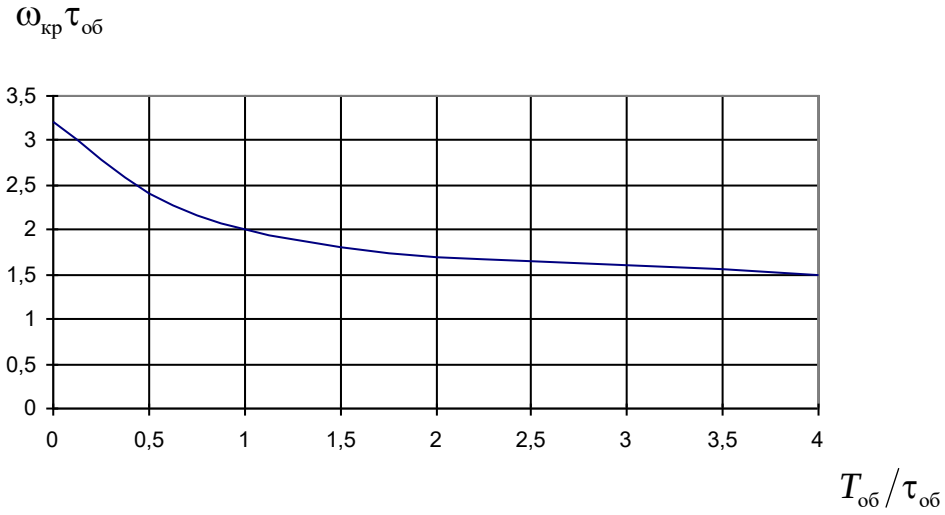


Рис. 2.35. Кривая предварительного выбора критической частоты

Этап 2. Вычисление на частоте $\omega_{кр}$ значения амплитудной характеристики $A_{об}(\omega_{кр})$ и критической настройки $C_1^{кр}$ по формуле (2.31).

Этап 3. Вычисление настроек регулятора по формулам:

а) для П-регулятора

$$C_1^* = 0,5C_1^{кр};\qquad (2.34)$$

б) для ПИ-регулятора

$$C_1^* = 0,45C_1^{кр};\qquad (2.35)$$

$$C_0^* = 0,086C_1^{кр}\omega_{кр};\qquad (2.36)$$

в) для ПИД-регулятора

$$C_1^* = 0,6C_1^{кр};\qquad (2.37)$$

$$C_0^* = 0,192C_1^{кр}\omega_{кр};\qquad (2.38)$$

$$C_2^* = 0,471\frac{C_1^{кр}}{\omega_{кр}}.\qquad (2.39)$$

Недостатком метода Циглера-Никольса является то, что он обеспечивает лишь один тип переходного процесса – с затуханием, равным ψ = 0,8 – 0,9.

Другим недостатком метода Циглера-Никольса является ограниченный набор типов законов регулирования (П, ПИ, ПИД).

2.5.2.6. Моделирование одноконтурной САР в частотной области

Анализ САР в частотной области имеет следующие цели:

1) оценить устойчивость замкнутой САР по АФЧХ разомкнутой САР $W_{рс}(j\omega)$;
2) найти частоту первого нуля $\omega_{р1}$ вещественной частотной характеристики замкнутой САР, необходимую для построения переходного процесса.

Модуль *SIANRG* рассчитан на проверку устойчивости одноконтурной САР по критерию Найквиста, формулировка которого для одноконтурной САР со статическим ОУ гласит, что АФЧХ $W_{рс}(j\omega)$ устойчивой системы в разомкнутом состоянии не охватывает точку (-1, j0), т.е. пересекает вещественную полуось справа от этой точки.

При анализе системы в частотной области вычисления производятся по формулам частотных передаточных функций разомкнутой САР:

$$W_{рс}(j\omega) = W_{об}(j\omega)R(j\omega) = U(\omega) + jV(\omega);\qquad (2.40)$$

замкнутой САР при подаче задающего сигнала на вход g:

$$W_{yg}(j\omega) = \frac{W_{об}(j\omega)R(j\omega)}{1+W_{об}(j\omega)R(j\omega)} = \frac{W_{рс}(j\omega)}{1+W_{рс}(j\omega)}\qquad (2.41)$$

или

$$W_{yg}(j\omega) = A_{yg}(\omega)e^{j\varphi(\omega)} = P_{yg}(\omega) + jQ_{yg}(\omega); \qquad (2.42)$$

замкнутой САР при подаче возмущающего сигнала на вход *f*:

$$W_{yf}(j\omega) = \frac{W_{об}(j\omega)}{1 + W_{об}(j\omega)R(j\omega)} = \frac{W_{об}(j\omega)}{1 + W_{рс}(j\omega)} \qquad (2.43)$$

или

$$W_{yf}(j\omega) = A_{yf}(\omega)e^{j\varphi(\omega)} = P_{yf}(\omega) + jQ_{yf}(\omega). \qquad (2.44)$$

Программа вычисляет на частоте ω_i значения $W_{об}(j\omega_i)$, $R(j\omega_i)$ по формулам соответствующих ПФ, после чего эти значения подставляются в формулы (2.37), (2.38), (2.40). При таком подходе ПК не может рассчитать АФЧХ $R(j\omega)$ для $\omega = 0$, если в ПФ регулятора существует интегрирующая составляющая. В этом случае необходимо использовать предельные переходы

$$\lim_{\omega\to 0} W_{yg}(j\omega) = 1; \qquad \lim_{\omega\to 0} W_{yf}(j\omega) = 0. \qquad (2.45)$$

Частоты ω_i в программе можно задавать произвольными дискретными значениями или диапазоном (ω_{min}, ω_{max}) с числом частот N_f внутри диапазона и видом интерполяции. ПМ обеспечивает линейную и логарифмическую интерполяции, задаваемые соответственно формулами

$$\omega_{i+1} = \omega_i + \frac{\omega_{max} - \omega_{min}}{N_f - 1}; \quad \omega_{i+1} = \omega_i 10^{(\omega_{max} - \omega_{min})/(N_f - 1)}.$$

Параметры частотного диапазона в ПМ вводятся в режиме диалога, причем возможна многократная модификация всех данных. Логарифмическая интерполяция применяется при исследовании АФЧХ в значительном диапазоне частот ($\omega_{min}/\omega_{max} > 100$).

2.5.2.7. Моделирование одноконтурной САР во временной области

После исследования САР на устойчивость необходимо оценить качество устойчивой САР. Для этого следует построить переходный процесс, вызванный единичным скачком по каналам задания и возмущения.

Вычисление переходного процесса в программном модуле основано на формулах численного интегрирования.

Интервал времени, на котором производится расчёт $h(t)$, определяется введенными в программу пользователем значениями начала (T_{min}) и конца (T_{max}) временного интервала. Этот интервал и количество точек переходного процесса N_f необходимо выбирать в интерактивном режиме, чтобы не потерять характерные колебания переходного процесса на начальных участках. Наибольшее значение T_{max} следует задавать до $10\tau_{об} - 20\tau_{об}$.

2.5.3. Описание работы программного модуля SIANRG

Моделирование в программном модуле *SIANRG* начинается с *Главного меню* (рис. 2.36), в котором имеются основные меню: *Файл*, *Синтез*, *Анализ*, *Отчеты*, *Помощь.*

Панель инструментов содержит кнопки для аналогичной основной работы пользователя: *Создать, Открыть, Сохранить, Синтез, Анализ, Помощь*.

Меню *Файл* содержит традиционные опции для работы с данными в виде файлов в формате **.dat*: *Создать, Открыть, Сохранить, Сохранить как…, Закрыть*.

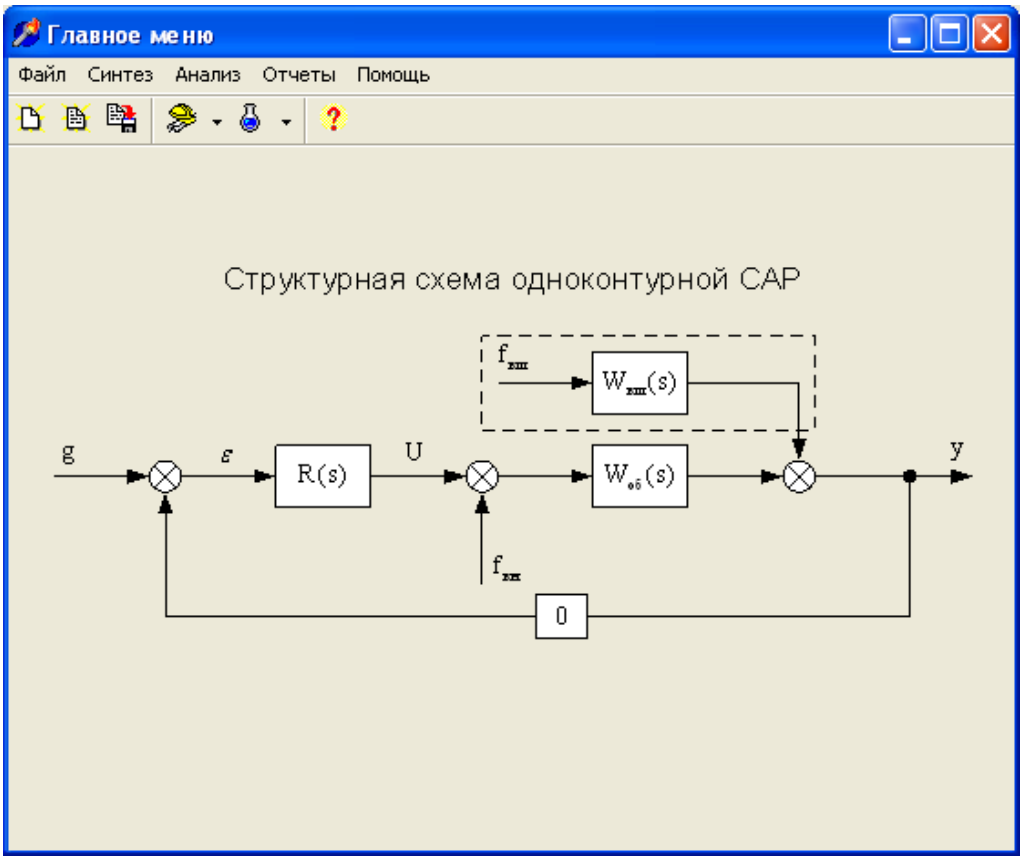


Рис. 2.36. Главное меню программного модуля *SIANRG*

Меню *Синтез* (см. рис. 2.37) содержит опции, позволяющие последовательно проводить синтез САР: задание параметров объекта (см. рис. 2.38, 2.39), расчет критической настройки регулятора (см. рис. 2.40, 2.41, 2.42), расчет оптимальных параметров регуляторов (см. рис. 2.43).

Для задания параметров ПФ ОУ выберите в главном меню пункт: *Синтез* / *Параметры объекта*. Перемещаться между компонентами можно с помощью клавиши *Tab* или с помощью мыши (подвести курсор к нужному компоненту и нажать левую клавишу мыши).

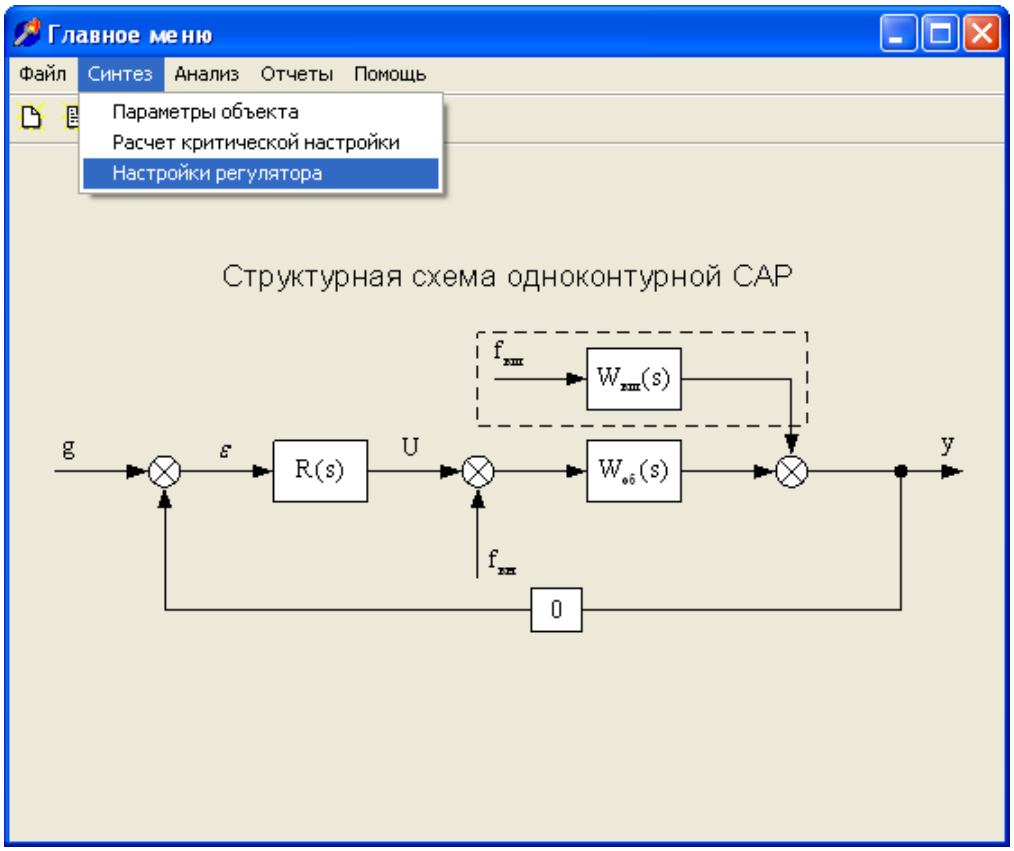


Рис. 2.37. Меню *Синтез САР*

Этапы задания передаточной функции объекта управления:

1. Задайте коэффициент усиления передаточной функции (ПФ) объекта управления (ОУ) $k_{об}$.

2. Задайте коэффициент транспортного запаздывания ПФ ОУ $\tau_{об}$.

3. Задайте числитель ПФ ОУ (см. рис. 2.38). Для этого выберите тип задания параметров числителя. Если необходимо задать два и больше параметров, то нажмите на кнопку *Добавить* столько раз сколько имеется параметров за минусом 1, а затем, выбирая параметр с нужным порядковым номером, задаются его значения (если тип задания параметров задан корнями, то записывается вещественная и мнимая части корня, в других случаях – только вещественная часть параметра – постоянная времени или коэффициент полинома). При моделировании простейшей одноконтурной САР числитель ПФ задается коэффициентом полинома, равным 1, номер параметра равен 0.

4. Если добавлен лишний параметр или необходимо отредактировать ПФ для меньшего числа параметров, выберите подлежащий удалению порядковый номер параметра и нажмите кнопку *Удалить* (удаляется текущий параметр).

5. Задайте знаменатель ПФ ОУ (см. рис. 2.39), руководствуясь пунктами 3 и 4. При моделировании простейшей одноконтурной САР знаменатель ПФ ОУ задается постоянной времени объекта $T_{об}$, номер параметра равен 0.

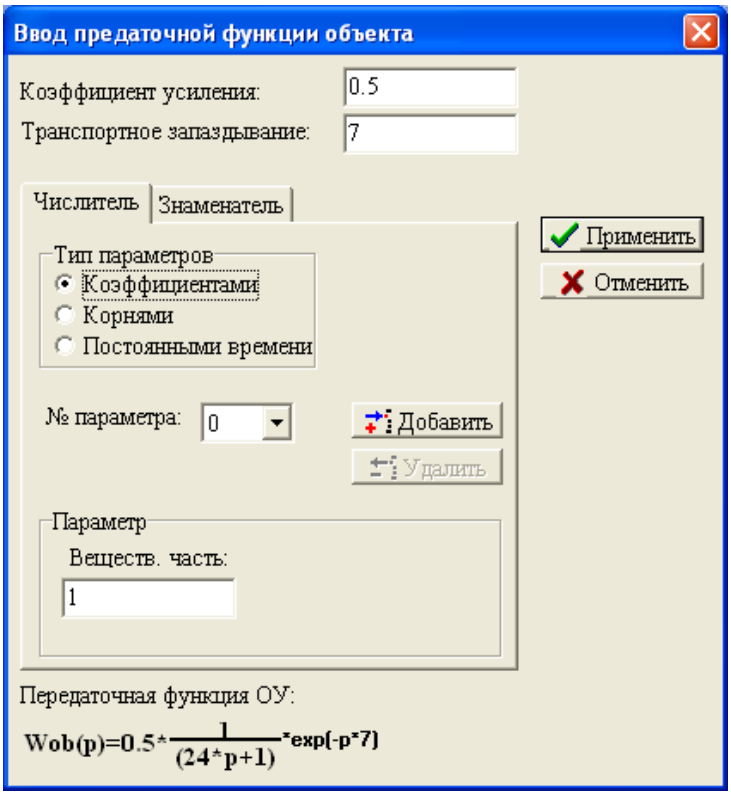


Рис. 2.38. Задание параметров объекта – ввод ПФ ОУ (числителя)

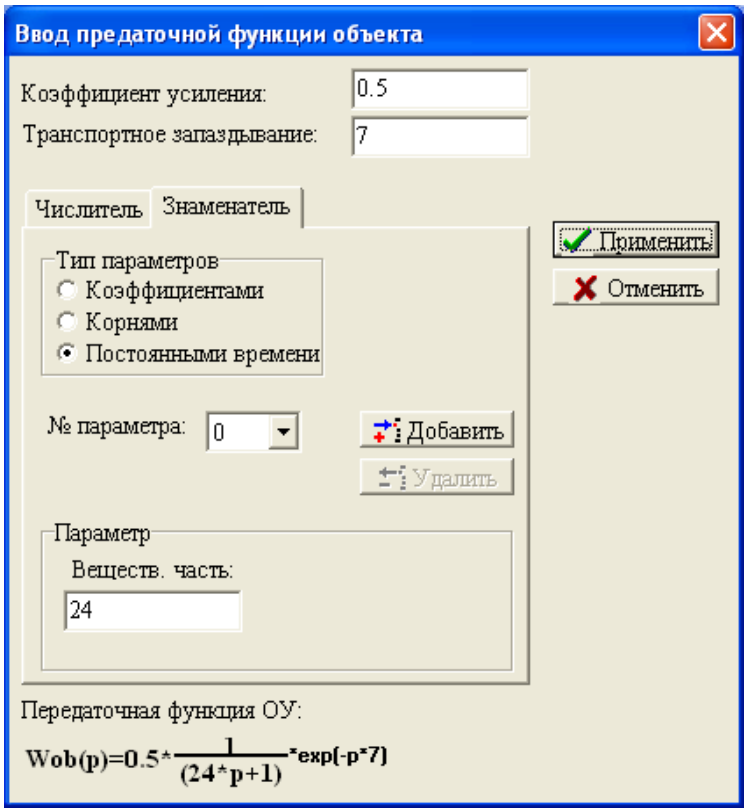


Рис. 2.39. Задание параметров объекта – ввод ПФ ОУ (знаменателя)

6. Все действия по заданию параметров сразу отображаются внизу окна, где представлена ПФ ОУ, соответствующая введенным данным (пункты 1, 2, 3 и 5).

7. Если все задано верно, то для запоминания введенных данных нажмите кнопку *Применить*, в противном случае – кнопку *Отмена* (это приведет к потере изменений в данных, т.е. ПФ ОУ не изменится).

Расчет критической настройки регулятора осуществляется в режиме интерактивного взаимодействия пользователя с программой в несколько этапов. На этапе поиска (рис. 2.40) предварительно оценивается критическая частота $\omega_{кр}$. Рекомендуется начинать производить оценку с задания нижней частоты $\omega_{н}$, равной 0,1, верхней частоты $\omega_{в}$ – 0,2; далее нажмите кнопку *Расчет*. Если при этом критическая частота не попадает в нужный диапазон ((2.32) – (2.33) и см. рис. 2.34), то на экране появится сообщение (рис. 2.41). Далее с шагом 0,1 задается снова нижняя (например, 0,2) и верхняя частоты (например, 0,3), (обязательно нажимайте кнопку *Расчет*) и т.д. до тех пор, пока не будет определена критическая частота и критическая настройка регулятора (см. рис. 2.42).

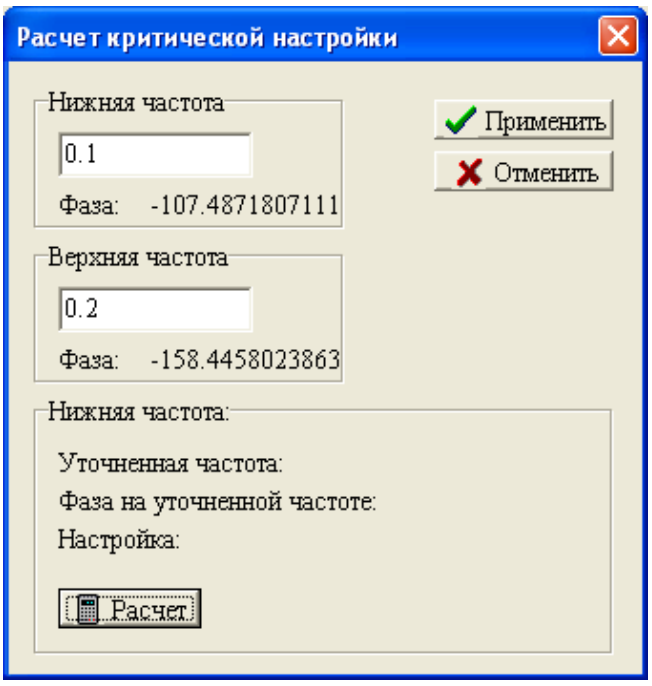


Рис. 2.40. Расчет критической настройки регулятора: этап поиска

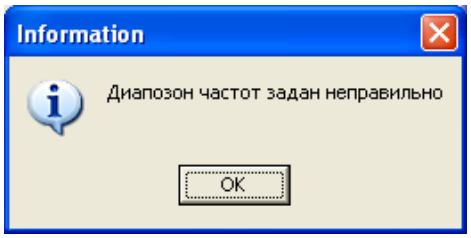


Рис. 2.41. Расчет критической настройки регулятора: информация пользователю

Для запоминания рассчитанных данных нажмите кнопку *Применить*, в противном случае – кнопку *Отмена* (это приведет к потере вновь рассчитанных данных, т.е. критическая настройка регулятора не изменится).

Расчет оптимальных параметров регулятора (рис. 2.43) можно произвести для формы 1 (2.10) – (2.12.) или формы 2 (2.15) – (2.17) для каждого регулятора (П-, ПИ-, ПИД-) отдельно. Выбрав форму задания параметров регулятора и вид регулятора, необходимо нажать кнопку *Оптимальные настройки*. Для запоминания рассчитанных данных нажмите кнопку *Применить*, в противном случае – кнопку *Отмена* (это приведет к потере вновь рассчитанных данных, т.е. оптимальные настройки регулятора не изменятся).

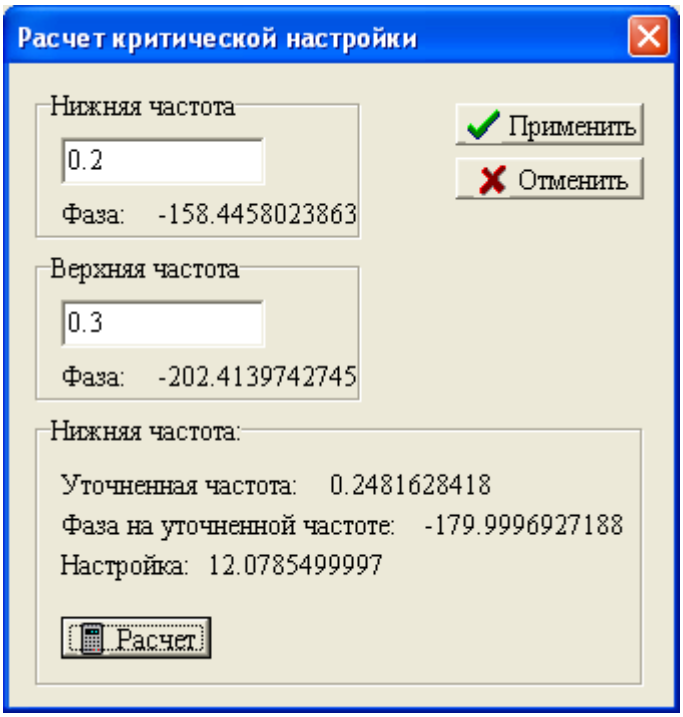


Рис. 2.42. Расчет критической настройки регулятора: заключительный этап

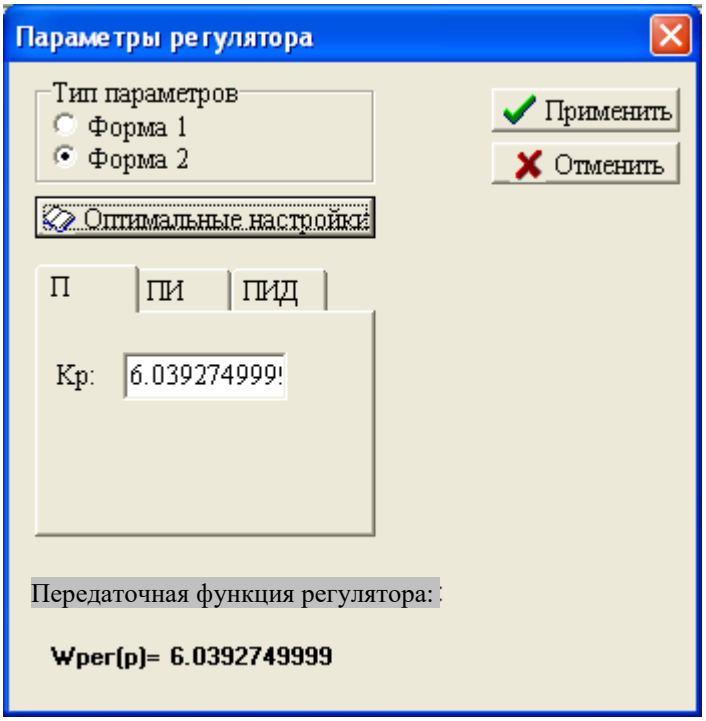


Рис. 2.43. Расчет оптимальных параметров П-регулятора

Меню *Анализ* (рис. 2.44) содержит опции, позволяющие последовательно проводить анализ: ОУ, разомкнутой САР в частотной и временной областях, замкнутой САР по управлению и возмущению (также в частотной и временной областях).

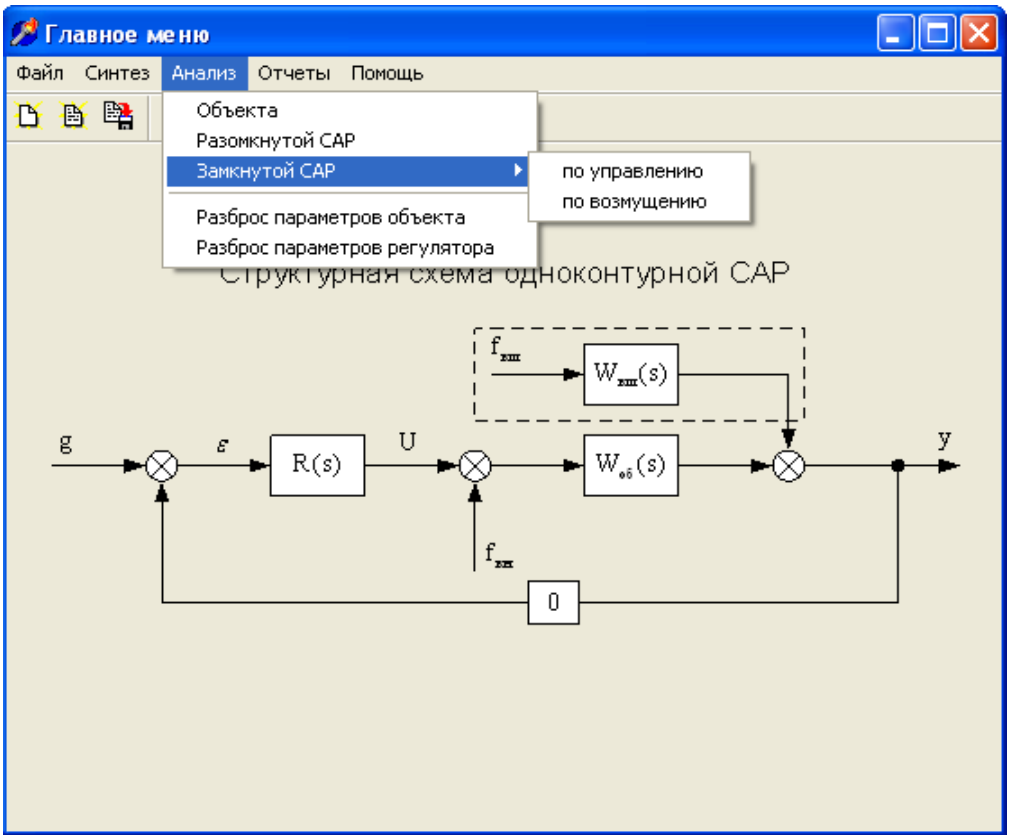


Рис. 2.44. Меню *Анализ*

В данном параграфе рассмотрим анализ:

– разомкнутой САР в частотной области;

– замкнутой САР по управлению во временной области;

– замкнутой САР по возмущению во временной области.

Анализ устойчивости разомкнутой САР в частотной области (см. рис. 2.45) осуществляется по АФЧХ согласно критерию Найквиста. Необходимо так задать настройки нижней и верхней частот, чтобы просмотреть АФЧХ системы в разомкнутом состоянии в районе точки (-1, j0). Напоминаем, что если АФЧХ пересекает вещественную полуось справа от этой точки, то система устойчива. После задания настроек необходимо нажать кнопку *Расчет*. Во время анализа устойчивости разомкнутой САР в частотной области необходимо сразу сохранять график АФЧХ в формате **.grs* (см. рис. 2.46), это позволит в дальнейшем правильно оформить отчет с наложением графиков. При анализе устойчивости разомкнутой САР в частотной области не рекомендуется менять настройки нижней и верхней частот, иначе в дальнейшем могут возникнуть трудности с наложением графиков.

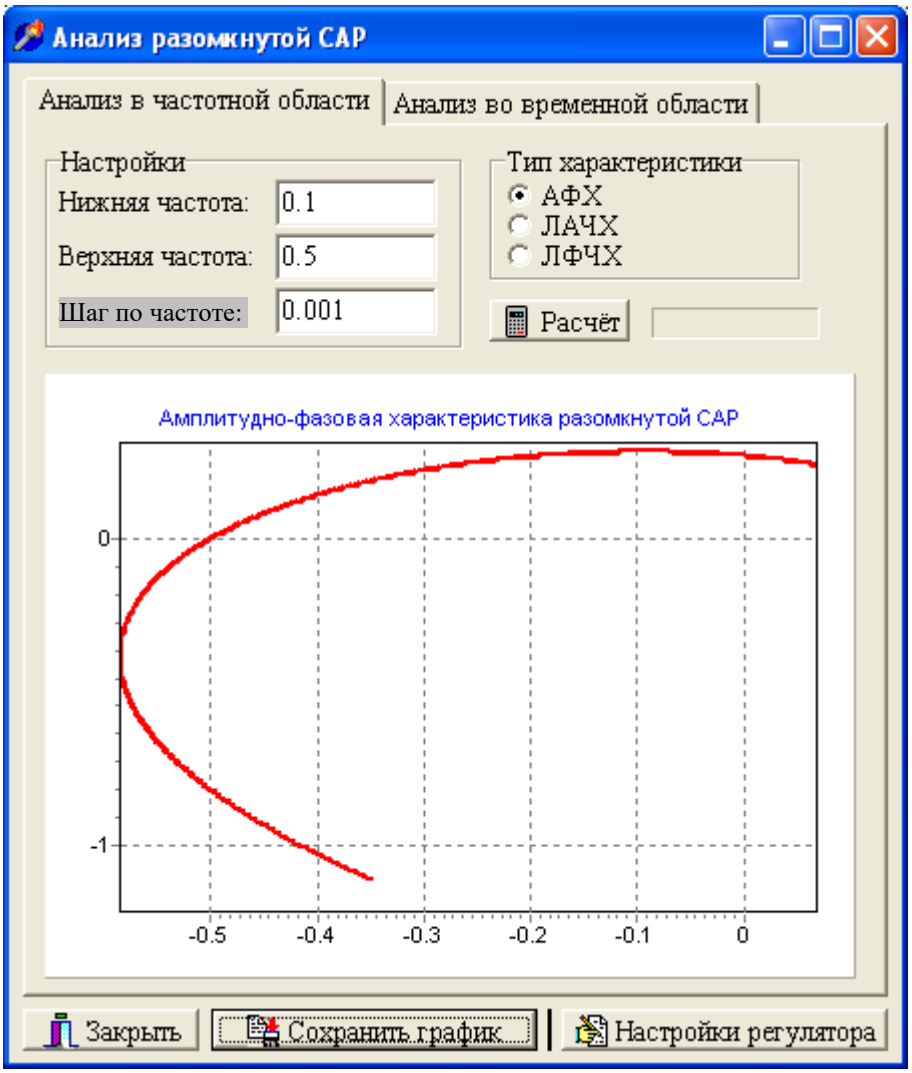


Рис. 2.45. Анализ разомкнутой САР в частотной области

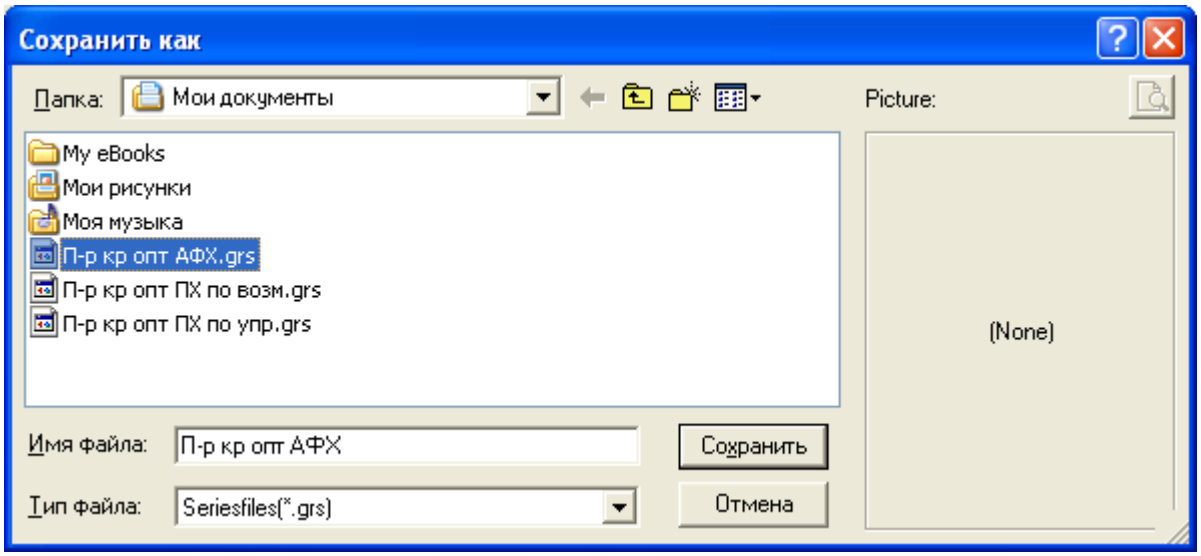


Рис. 2.46. Сохранение графика АФЧХ разомкнутой САР при анализе в частотной области

Анализ качества функционирования замкнутой САР во временной области по управлению (см. рис. 2.47) осуществляется по ПХ. Необходимо так задать настройки начального и конечного времени и шага по времени, чтобы просмотреть ПХ системы до момента полного установившегося режима. Уставка задается равной 1, поскольку ПХ $h(t)$ – это есть реакция системы на типовое воздействие в виде единичной ступенчатой функции $1(t)$. После задания настроек необходимо нажать кнопку *Расчет*. Во время анализа качества функционирования замкнутой САР во временной области по управлению необходимо сразу сохранять график ПХ в формате *.*grs* (см. рис. 2.48), это позволит в дальнейшем правильно оформить отчет с наложением графиков. При анализе качества функционирования замкнутой САР также не рекомендуется менять настройки начального и конечного времени и шага по времени.

Анализ замкнутой САР во временной области по возмущению (см. рис. 2.49) также осуществляется по ПХ, с сохранением графика (см. рис. 2.50).

Анализируя устойчивость и качество функционирования САР, необходимо изменять настройки регулятора. Для этого в окнах *Анализа САР* (см. рис. 2.45, 2.47, 2.49) достаточно нажать кнопку *Настройки регулятора,* и пользователь попадает в окно *Параметры регулятора* (см. рис. 2.43). Далее, выбирая тип регулятора, необходимо согласно плану однофакторного эксперимента изменить соответствующую настройку, т.е. ввести новое ее значение (см. рис. 2.51, *а*) и обязательно нажать кнопку *Применить* (см. рис. 2.51, *б*), иначе в памяти программы остается старое значение (см. рис. 2.51, *а*).

После этого программа автоматически возвращается в окно *Анализа САР*, в котором, нажав кнопку *Расчет*, не меняя настройки для просмотра графиков, можно получить новые результаты анализа соответствующей САР. Не забывайте сразу сохранять графики.

Для эффективности проведения однофакторного эксперимента рекомендуется, задав новое значение настройки регулятора, например для системы с П-регулятором $k_{\text{р}}^{\text{о}} + \Delta k_{\text{р}}$, проанализировать САР во всех трех вариантах: разомкнутая САР в частотной области (см. рис. 2.52), замкнутая САР во временной области по управлению (см. рис. 2.53) и замкнутая САР во временной области по возмущению (см. рис. 2.5). Сохраните графики в формате *.*grs*. Затем задайте значение настройки для системы с П-регулятором $k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}}$ (см. рис. 2.55) и также проанализируйте САР во всех трех вариантах: разомкнутая САР в частотной области (см. рис. 2.56), замкнутая САР во временной области по управлению (см. рис. 2.57) и замкнутая САР во временной области по возмущению (см. рис. 2.58). Сохраните графики в формате *.*grs*.

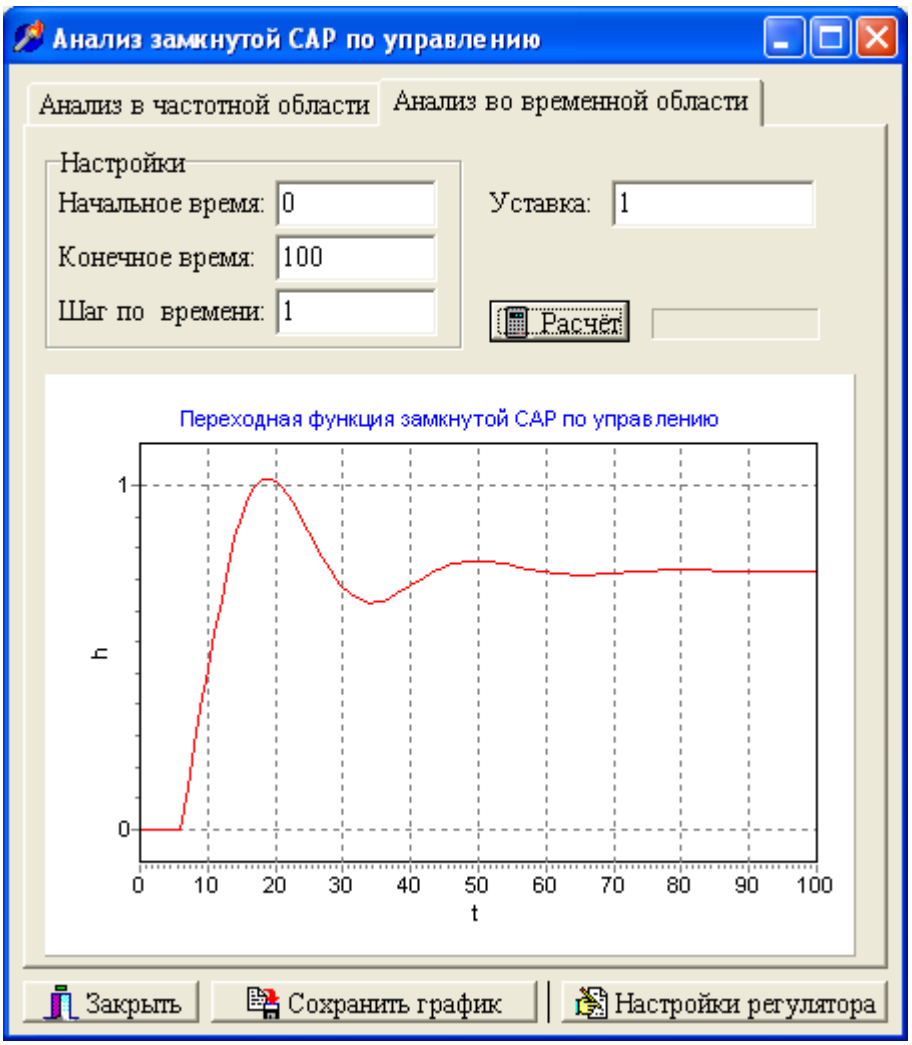


Рис. 2.47. Анализ замкнутой САР во временной области по управлению

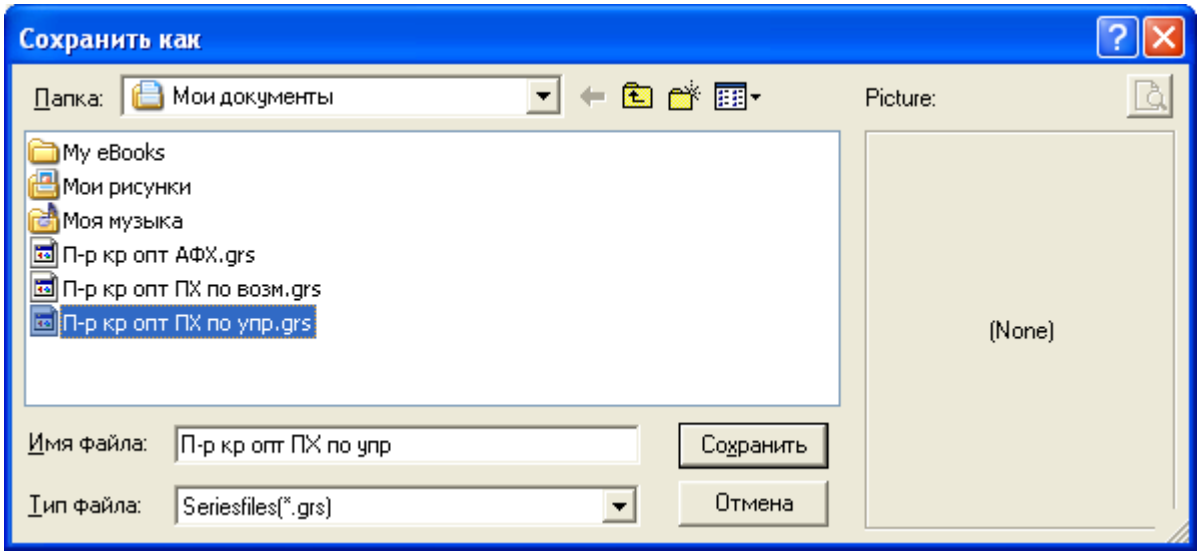


Рис. 2.48. Сохранение графика ПХ замкнутой САР при анализе
во временной области по управлению

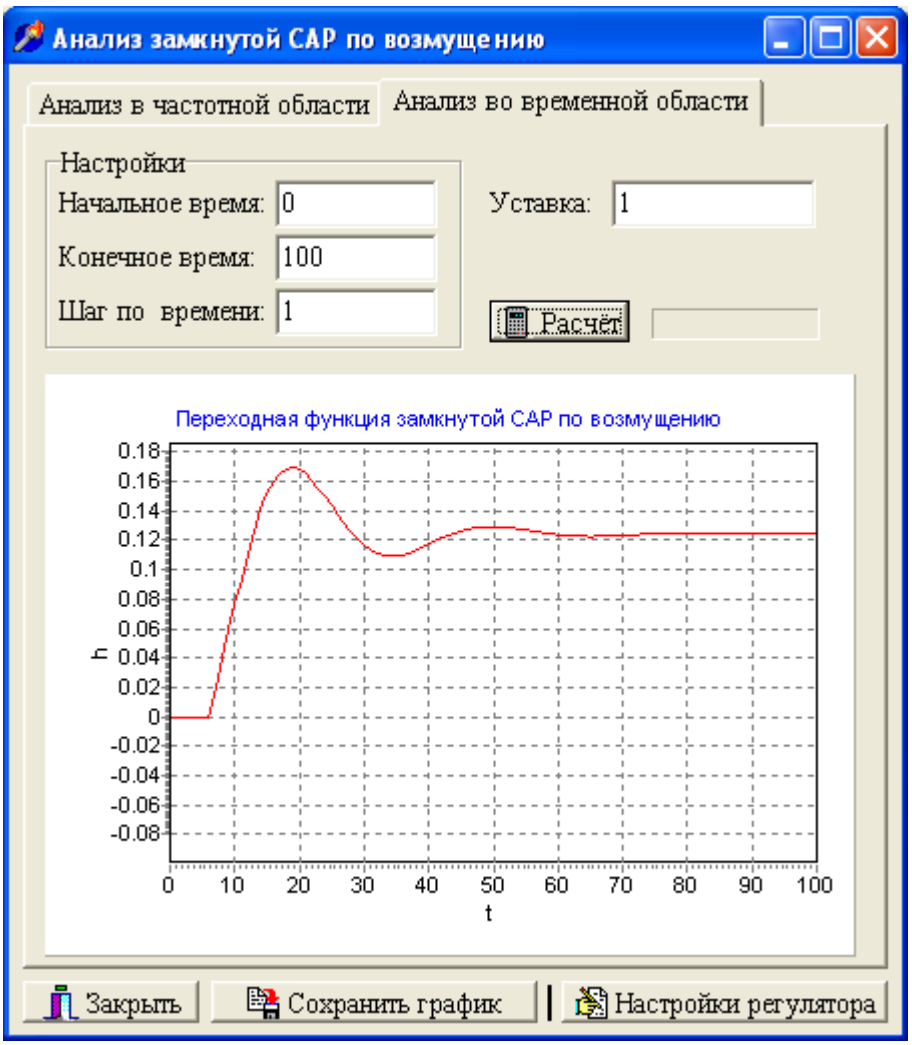


Рис. 2.49. Анализ замкнутой САР во временной области по возмущению

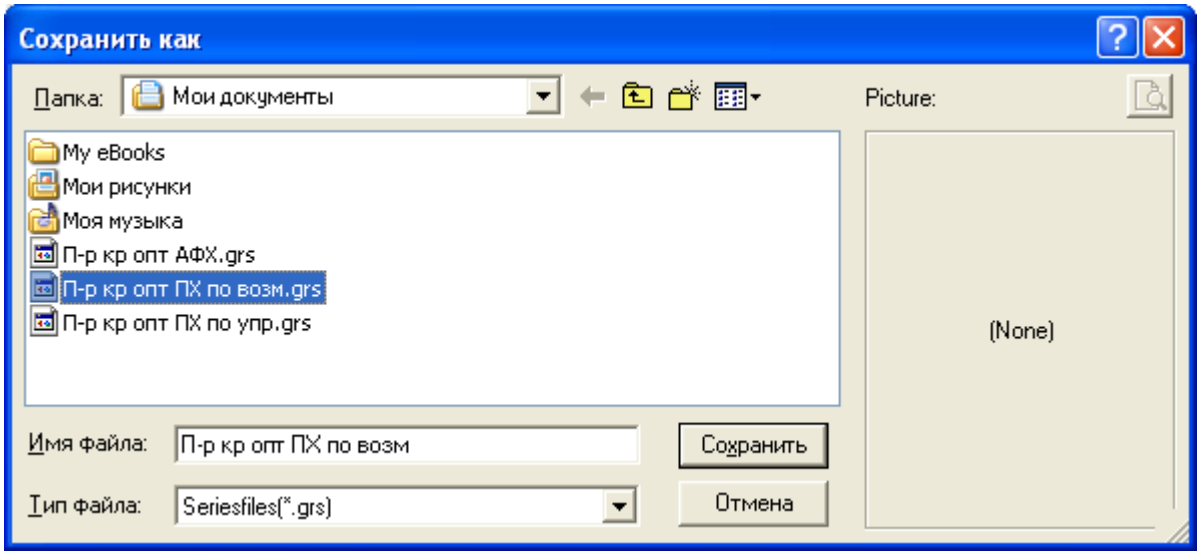


Рис. 2.50. Сохранение графика ПХ замкнутой САР при анализе во временной области по возмущению

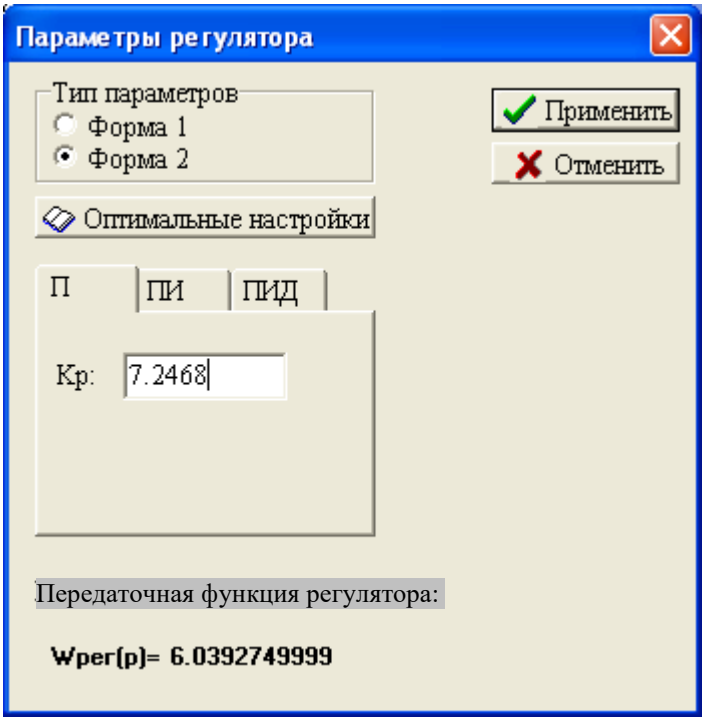


Рис. 2.51, *а*. Изменение настройки регулятора: ввод нового значения

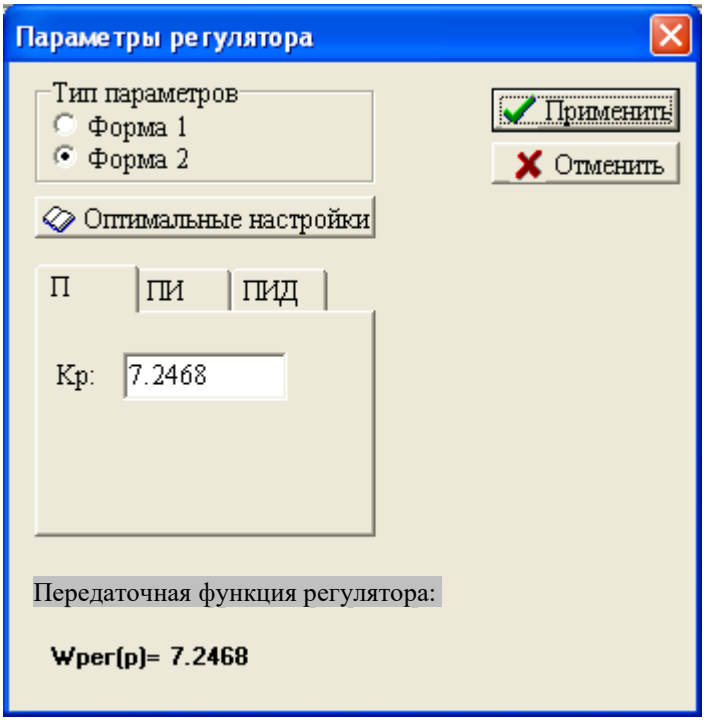


Рис. 2.51, *б*. Изменение настройки регулятора: после нажатия кнопки *Применить*

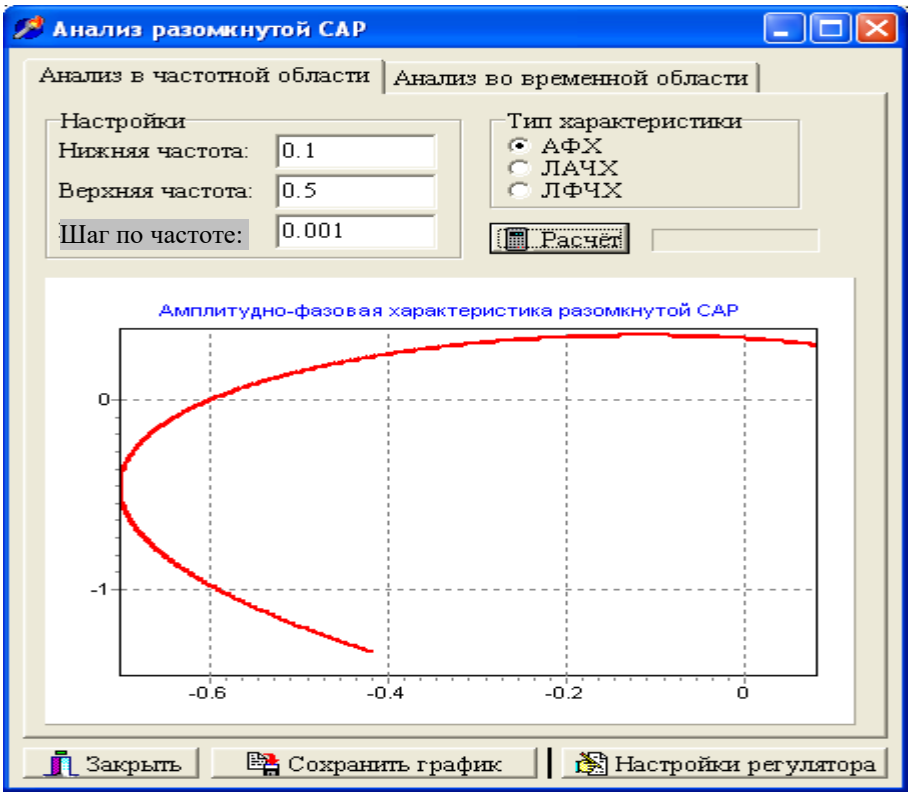


Рис. 2.52. Анализ разомкнутой САР в частотной области для системы с П-регулятором при $k_{\text{р}}^{\text{о}} + \Delta k_{\text{р}}$

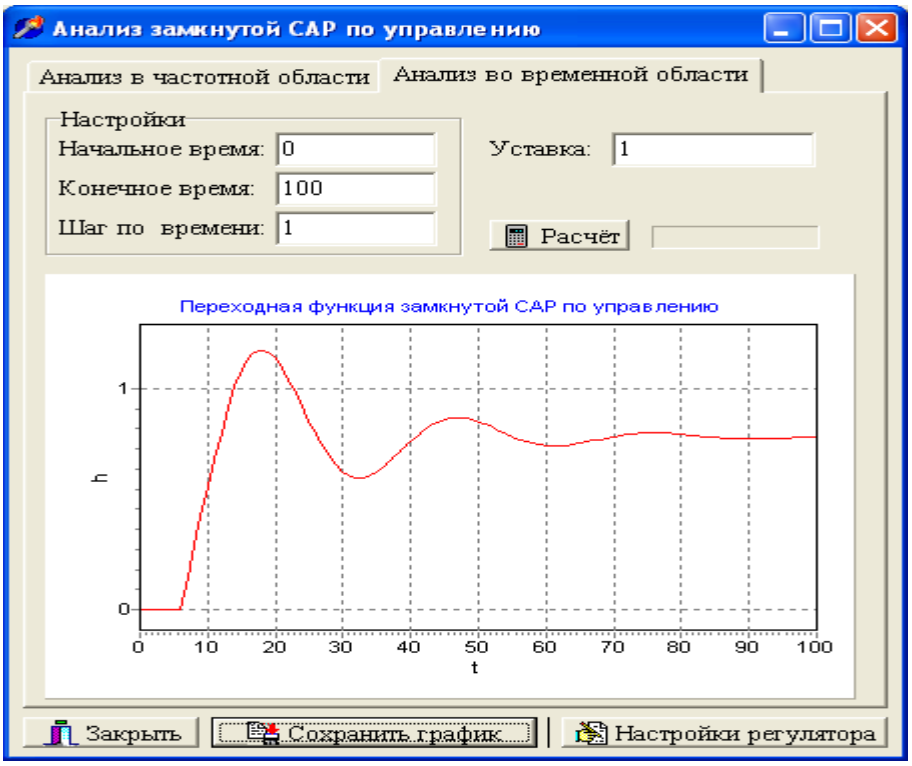


Рис. 2.53. Анализ замкнутой САР во временной области по управлению для системы с П-регулятором при $k_{\text{р}}^{\text{о}} + \Delta k_{\text{р}}$

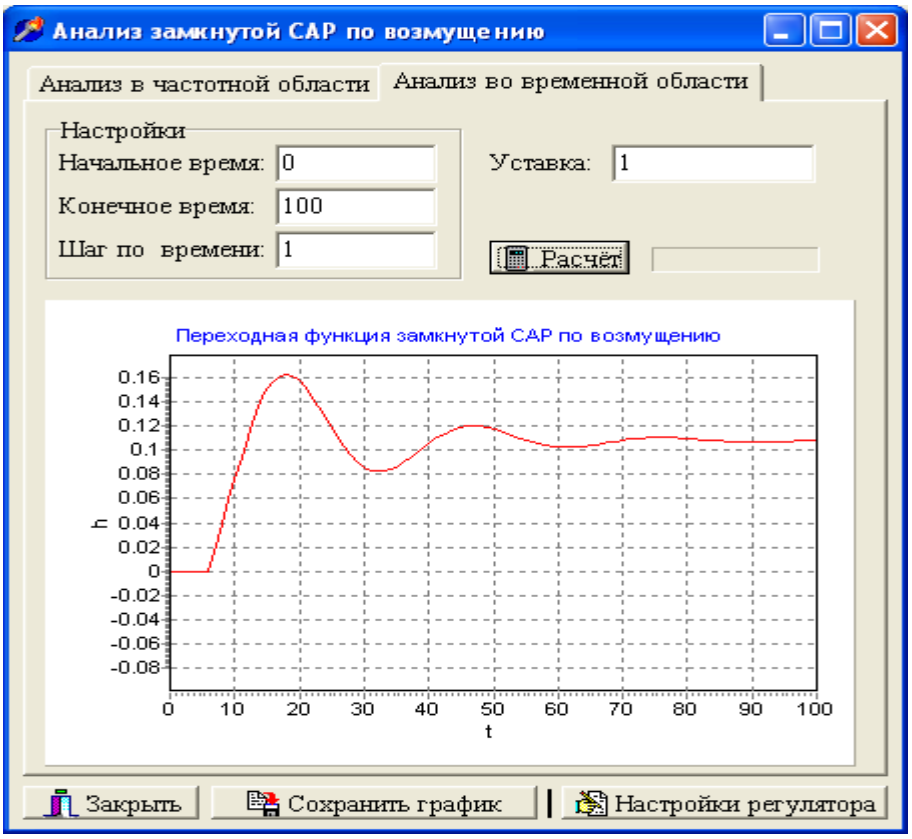


Рис. 2.54. Анализ замкнутой САР во временной области по возмущению для системы с П-регулятором при $k_{\text{р}}^{\text{о}} + \Delta k_{\text{р}}$

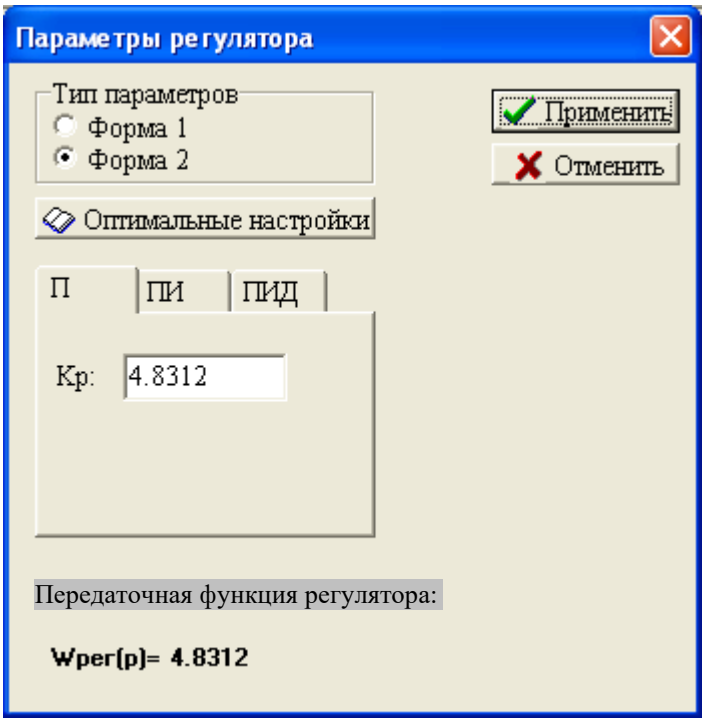


Рис. 2.55. Новое значение настройки для САР с П-регулятором $k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}}$

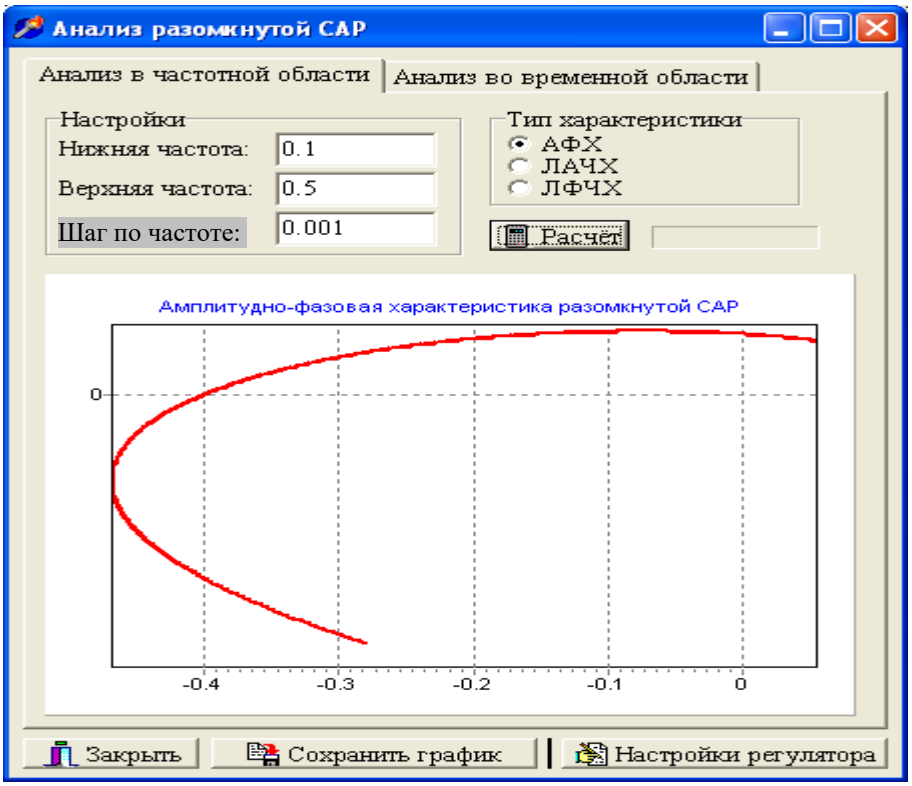


Рис. 2.56. Анализ разомкнутой САР в частотной области для системы с П-регулятором при $k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}}$

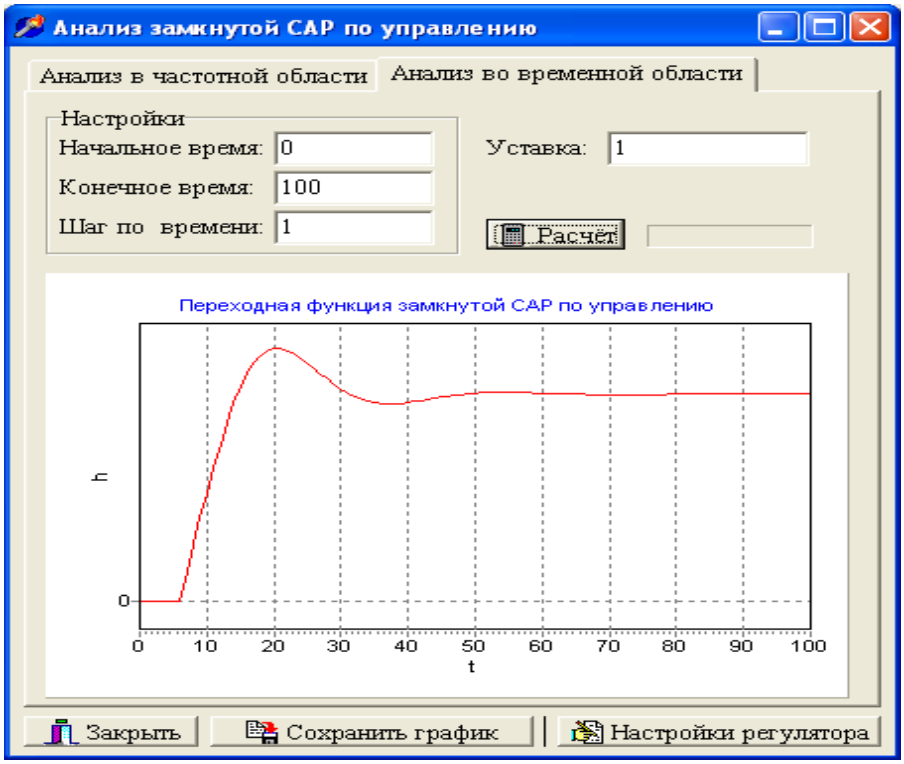


Рис. 2.57. Анализ замкнутой САР во временной области по управлению для системы с П-регулятором при $k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}}$

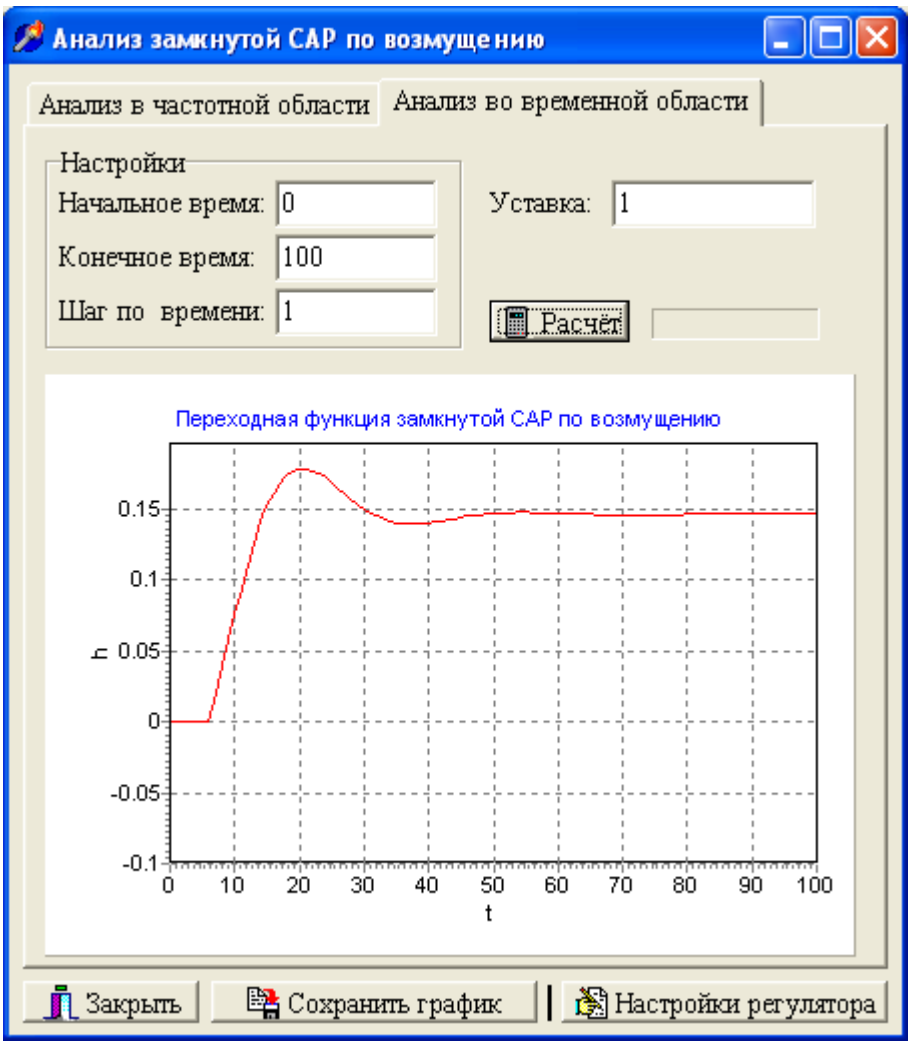


Рис. 2.58. Анализ замкнутой САР во временной области по возмущению для системы с П-регулятором при $k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}}$

Таким образом, получаются семейства характеристик: АФЧХ для разомкнутой САР по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$ (см. рис. 2.59); ПХ замкнутой САР по управлению по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$ (см. рис. 2.60); ПХ замкнутой САР по возмущению по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$ (см. рис. 2.61). Для представления результатов однофакторного эксперимента в виде семейства характеристик воспользуйтесь меню *Отчет / Наложение графиков*. Сначала выберите тип графика, затем, нажимая кнопку *Добавить*, вставьте соответствующий график. Если тип графика не соответствует, т.е. вы выбрали АФЧХ, а пытаетесь добавить график ПХ, то график не будет вставлен. Обязательно сразу сохраните каждое семейство графиков в формате, например **.bmp* (см. рис. 2.62). Для эффективности написания отчета по самостоятельной работе сразу подписывайте графики, а также указывайте названия и значения осей.

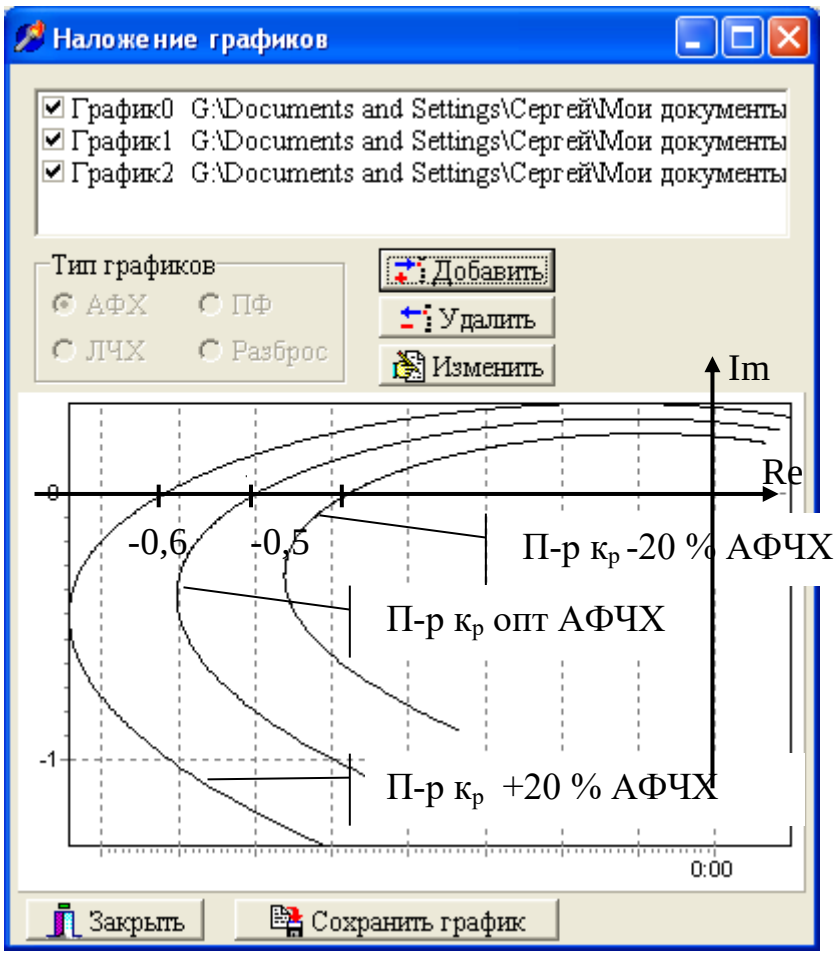


Рис. 2.59. Семейство АФЧХ для разомкнутой САР при $k_р^{o}$, $+\Delta k_р$, $-\Delta k_р$

В дальнейшем для составления нового семейства графиков необходимо сначала удалить предыдущие графики, нажимая кнопку *Удалить*, далее выбрать новый тип графика и, нажимая кнопку *Добавить*, вставить соответствующие графики.

Таким образом, однофакторный эксперимент для САР с П-регулятором завершен.

Далее необходимо проанализировать результаты эксперимента:

– по АФЧХ разомкнутой САР в частотной области оценить устойчивость системы;

– по ПХ замкнутой САР во временной области по управлению и возмущению определить параметры качества системы: время регулирования $t_р$ при 5%-ной δ-трубке (см. рис. 2.32, 2.33), максимальные динамические отклонения σ по (2.22) и (2.23), перерегулирование η по (2.24) и (2.25);

– выявить закономерности влияния варьируемых параметров регулятора на устойчивость и показатели качества системы.

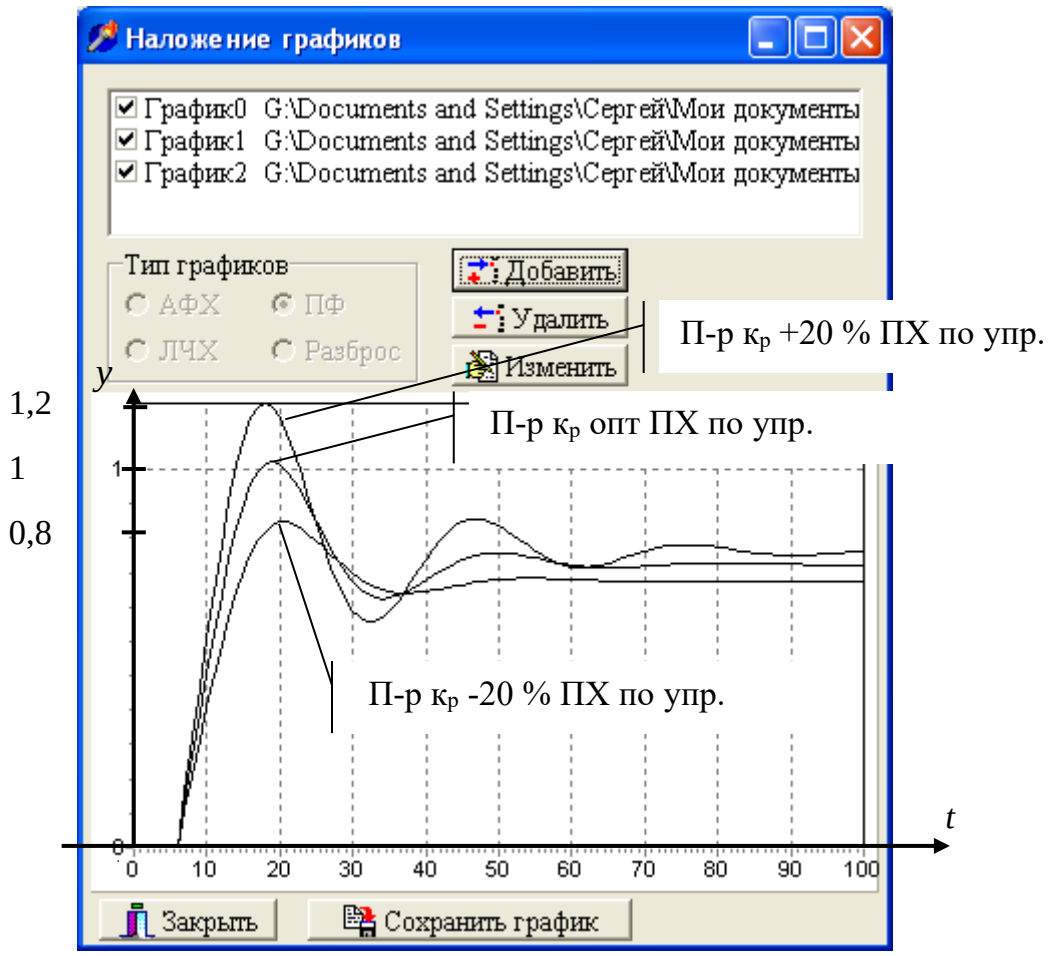


Рис. 2.60. Семейство ПХ для замкнутой САР по управлению при $k_{\text{р}}^{\text{о}}, \pm \Delta k_{\text{р}}$

Далее необходимо провести однофакторные эксперименты для САР с ПИ- и ПИД-регуляторами.

Для системы с ПИ-регулятором при проведении однофакторного эксперимента:

– сначала задайте оптимальные значения параметров регулятора (см. рис. 2.63);

– затем изменяйте коэффициент передачи регулятора $k_{\text{р}} \pm \Delta k_{\text{р}}$ при постоянных значениях $T_{\text{И}} = T_{\text{И}}^{\text{о}}$;

– после изменяйте постоянную интегрирования $T_{\text{И}} \pm \Delta T_{\text{И}}$ при постоянных значениях $k_{\text{р}} = k_{\text{р}}^{\text{о}}$.

Каждый раз анализируйте результаты эксперимента и выявите закономерности влияния варьируемых параметров регулятора на устойчивость и показатели качества системы.

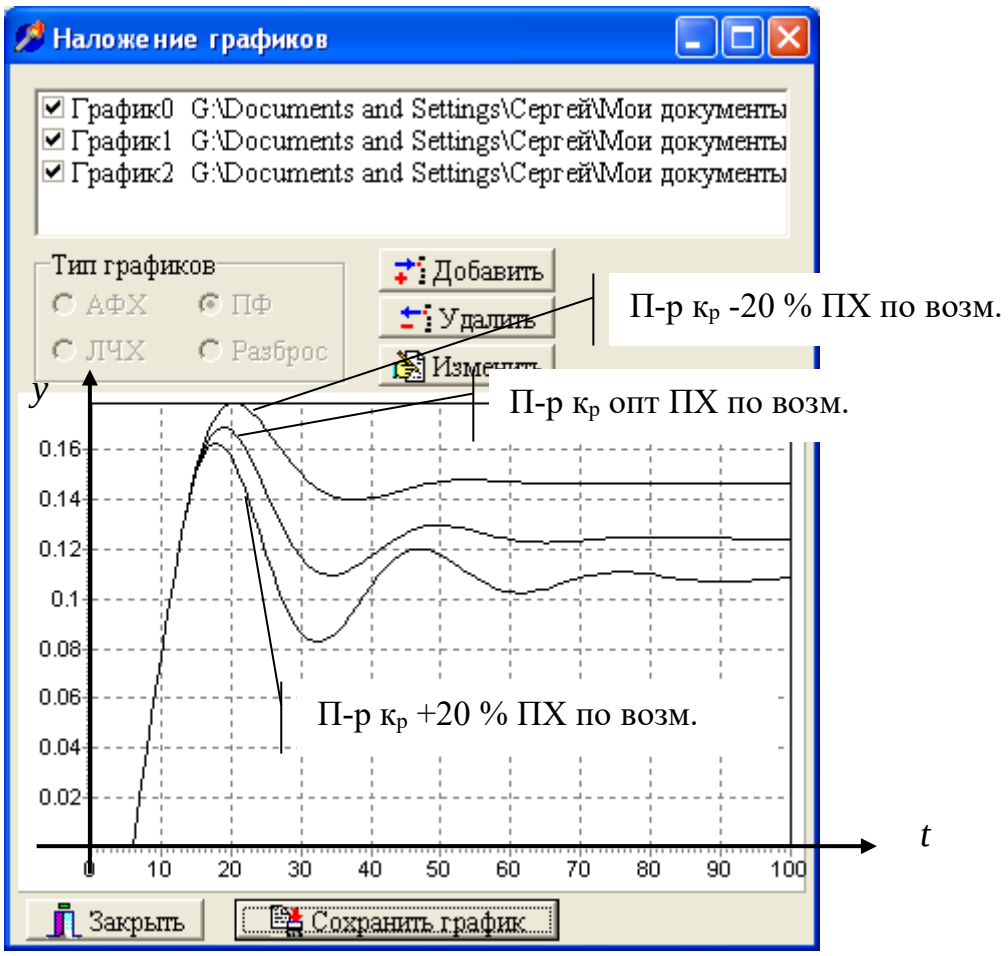


Рис. 2.61. Семейство ПХ для замкнутой САР по возмущению при $k_{\text{р}}^{\text{о}}$, $\pm \Delta k_{\text{р}}$

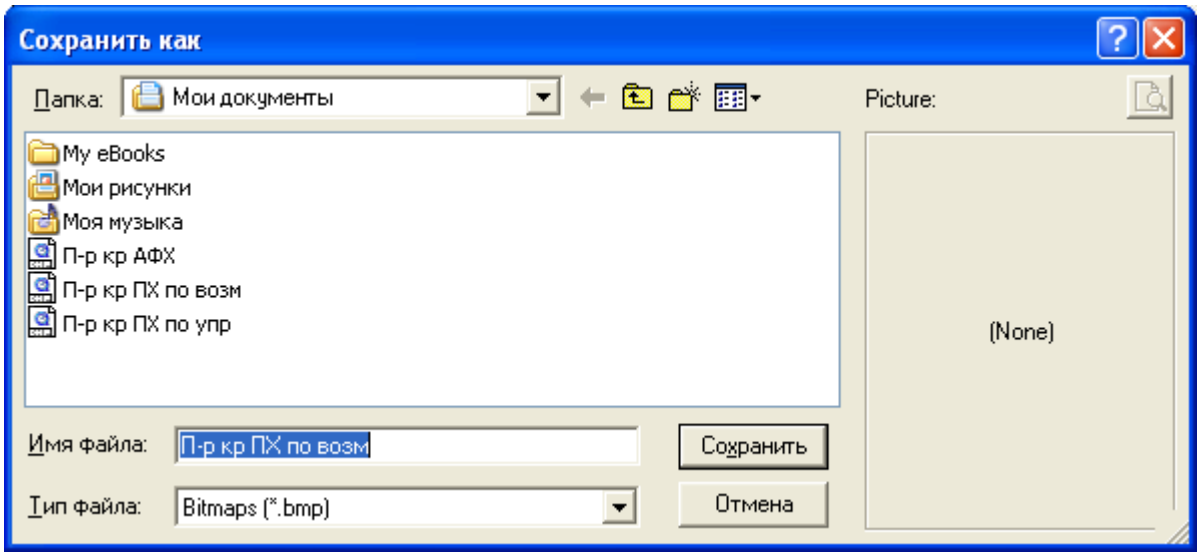


Рис. 2.62. Сохранение семейства графиков в формате *.*bmp*

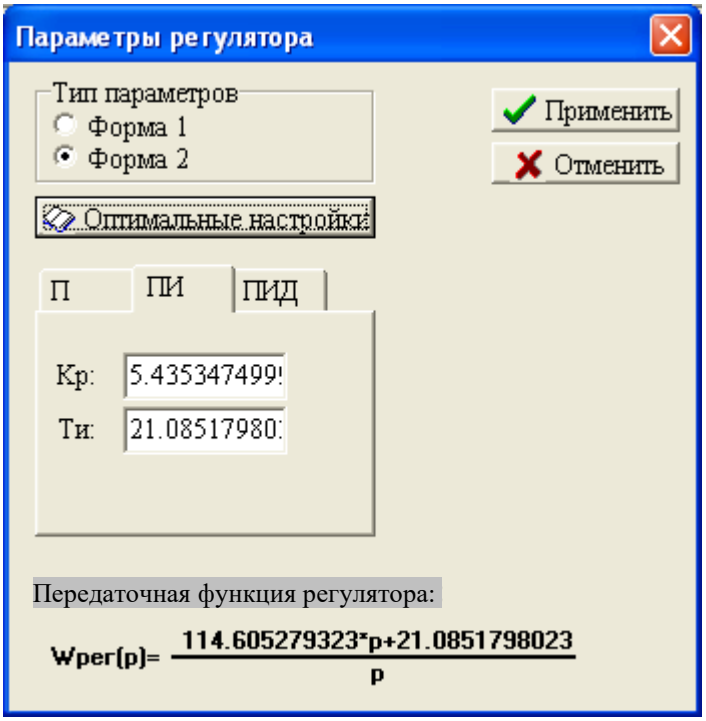


Рис. 2.63. Определение оптимальных значений параметров ПИ-регулятора

Для системы с ПИД-регулятором при проведении однофакторного эксперимента:

– сначала задайте оптимальные значения параметров регулятора (см. рис. 2.64);

– затем изменяйте коэффициент передачи регулятора $k_р \pm \Delta k_р$ при постоянных значениях $T_И = T_И^o$ и $T_Д = T_Д^o$;

– далее изменяйте постоянную интегрирования $T_И \pm \Delta T_И$ при постоянных значениях $k_р = k_р^o$ и $T_Д = T_Д^o$;

– и наконец изменяйте постоянную дифференцирования $T_Д \pm \Delta T_Д$ при постоянных значениях $k_р = k_р^o$ и $T_И = T_И^o$.

Каждый раз анализируйте результаты эксперимента и выявите закономерности влияния варьируемых параметров регулятора на устойчивость и показатели качества системы.

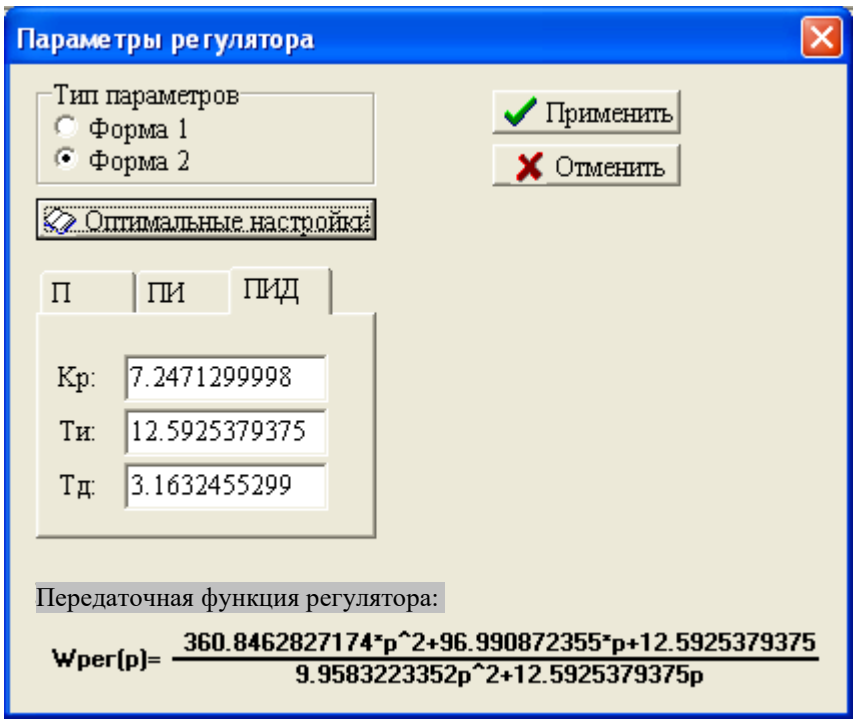


Рис. 2.64. Определение оптимальных значений параметров ПИД-регулятора

На рис. 2.65 представлен шаблон оформления отчета по самостоятельной работе с помощью меню *Отчеты*. В данном шаблоне при построении отчета достаточно выбрать из списка данные (по ОУ, регулятору, критические характеристики), файлы сохраненных графиков, вписать свои комментарии, предварительно просмотреть отчет и сохранить его в виде нового файла или дописать в существующий.

На рис. 2.66, *а – е* представлены информационные окна, которые появляются в случае неправильно заданных пользователем данных.

На рис. 2.67 представлено меню *Помощь*, в котором подробно описано окно *Ввода параметров ОУ*, последовательность заполнения ячеек данных, возможности сохранения (применения) или несохранения (отмены) вновь введенных данных.

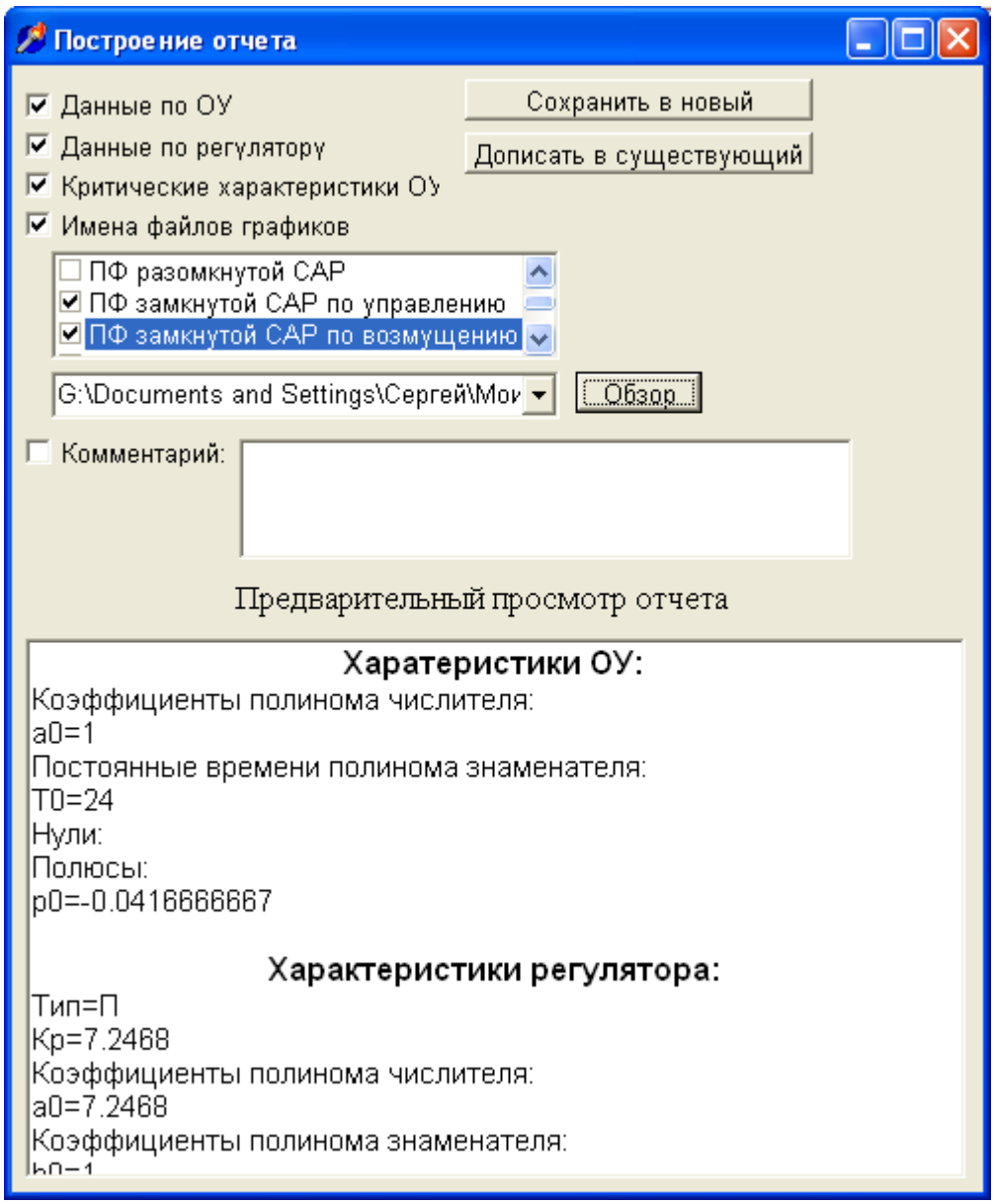
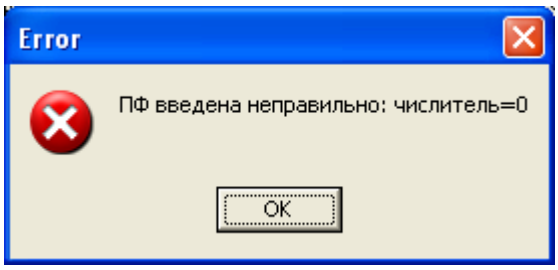
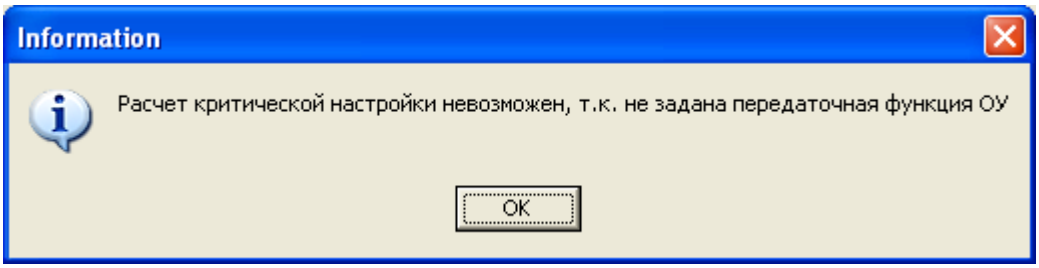


Рис. 2.65. Оформление отчета с помощью меню *Отчеты*

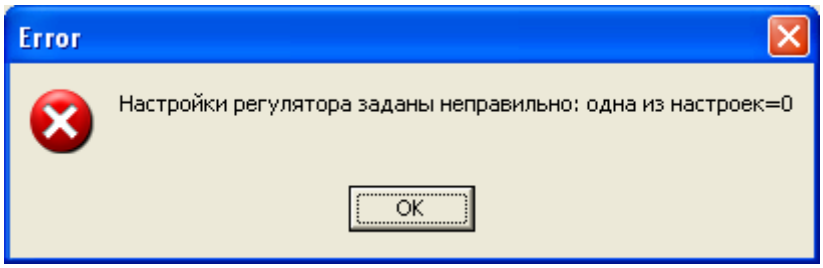
а



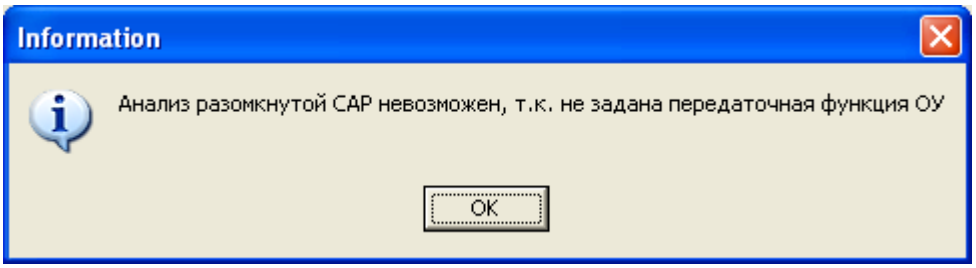
б



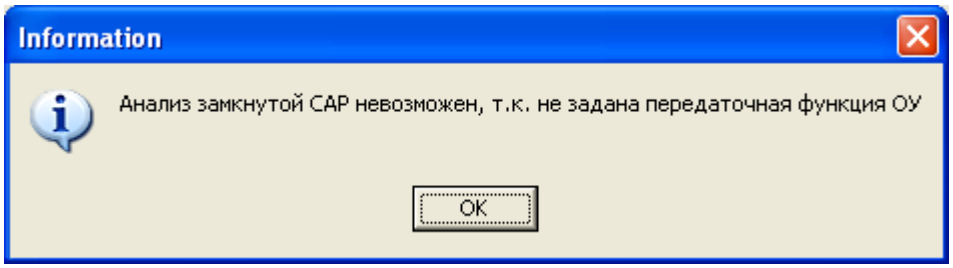
в



г



д



е

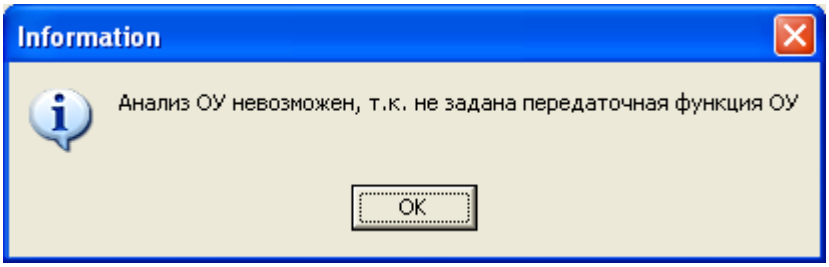


Рис. 2.66, *а* – *е*. Информационные окна

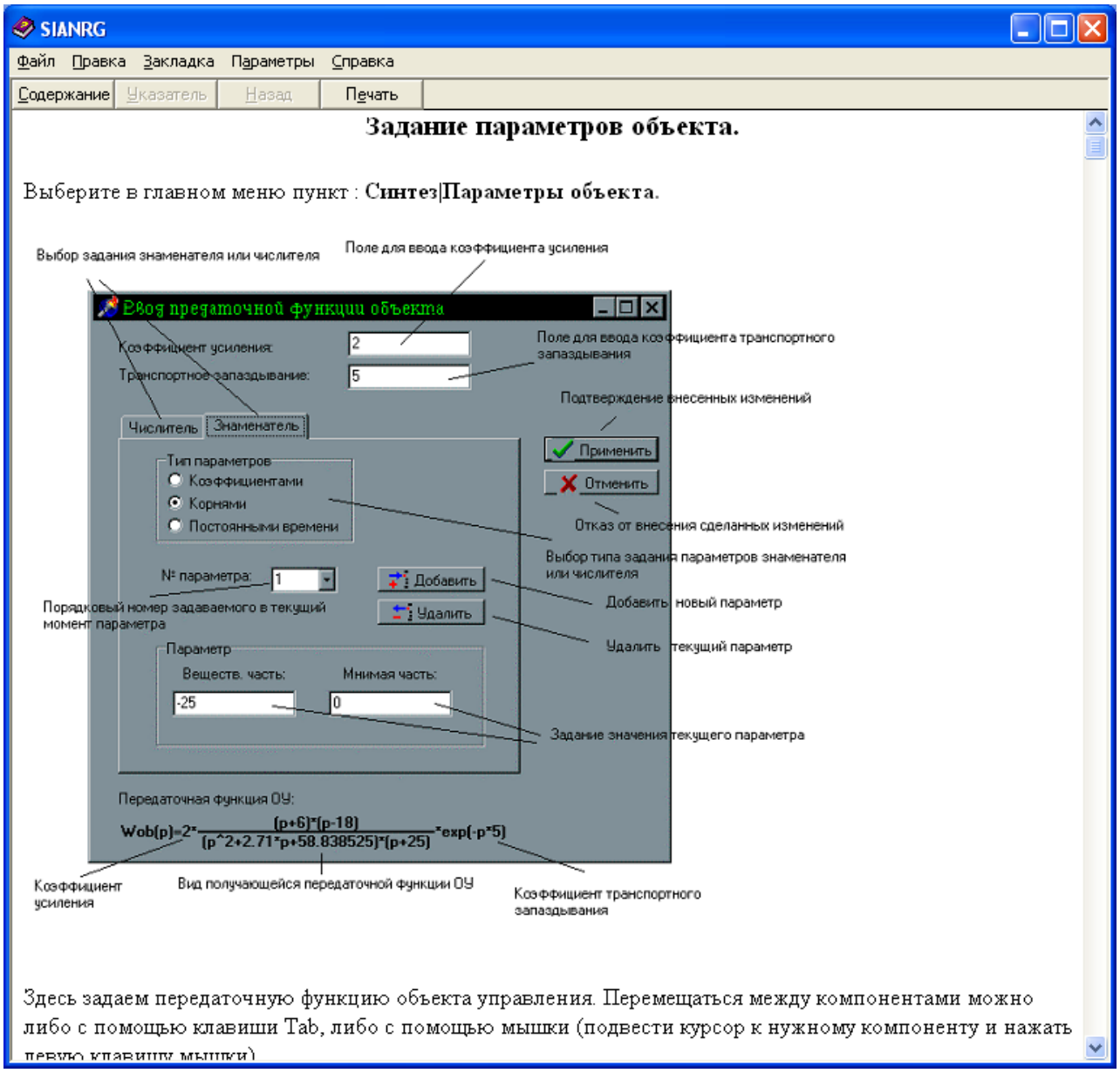


Рис. 2.67. Меню *Помощь*

2.5.4. Задания для самостоятельной работы

1. Запустить программу *C:\USERS\MS\Sianrg.exe*, которая используется для моделирования САР.

2. Составить параметрическую модель СУ, для чего следует найти базовые (оптимальные) настройки П-регулятора $k_{\text{р}}^{\text{о}}$; ПИ-регулятора $k_{\text{р}}^{\text{о}}$ и $T_{\text{И}}^{\text{о}}$ и для ПИД-регулятора $k_{\text{р}}^{\text{о}}$, $T_{\text{И}}^{\text{о}}$ и $T_{\text{Д}}^{\text{о}}$ методом Циглера-Никольса (см. разд. 2.5.2.5). Данные по передаточной функции (2.8) объекта управления $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ можно выбрать из табл. 2.8. Номер варианта уточнить у преподавателя.

3. Провести моделирование СУ в частотной области. Для этого вычислить АФЧХ разомкнутой СУ и оценить по ней устойчивость системы.

Таблица 2.8

Варианты задания

Номер вари-анта	$k_{об}$	$T_{об}$	$\tau_{об}$	Номер вари-анта	$k_{об}$	$T_{об}$	$\tau_{об}$
1	2	8	2,4	6	0,2	4	2,0
2	3	10	3,0	7	0,3	5	2,5
3	5	12	3,6	8	0,5	6	3,0
4	6	14	4,2	9	2	100	30
5	2	8	4,0	10	3	120	36

4. Провести моделирование СУ во временной области. Для этого определить переходные процессы системы по управлению и возмущению. Построить графики ПХ и определить время регулирования $t_р$ при 5%-ной δ-трубке (см. рис. 2.32, 2.33); максимальное динамическое отклонение σ по (2.22) и (2.23), перерегулирование η по (2.24) и (2.25). Далее эти параметры будут являться базовыми или оптимальными показателями качества СУ.

5. С целью определения устойчивости системы к детерминированным вариациям параметров регулятора провести однофакторный эксперимент [2.9]. В качестве факторов эксперимента выбрать $k_р$, $T_И$ и $T_Д$. Уровень вариации параметров регулятора Δ может быть равен ±10, ±20, ±30 % (уточнить у преподавателя):

а) для системы с П-регулятором необходимо изменять коэффициент передачи регулятора $k_р \pm \Delta k_р$;

б) для системы с ПИ-регулятором необходимо сначала изменять коэффициент передачи регулятора $k_р \pm \Delta k_р$ при постоянных значениях $T_И = T_И^о$; затем изменять постоянную интегрирования $T_И \pm \Delta T_И$ при постоянных значениях $k_р = k_р^о$;

в) для системы с ПИД-регулятором необходимо сначала изменять коэффициент передачи регулятора $k_р \pm \Delta k_р$ при постоянных значениях $T_И = T_И^о$ и $T_Д = T_Д^о$; затем изменять постоянную интегрирования $T_И \pm \Delta T_И$ при постоянных значениях $k_р = k_р^о$ и $T_Д = T_Д^о$; и, наконец, изменять постоянную дифференцирования $T_Д \pm \Delta T_Д$ при постоянных значениях $k_р = k_р^о$ и $T_И = T_И^о$.

По АФЧХ разомкнутой САР в частотной области оценить устойчивость системы, многократно повторяя пункт 3. Определить критическое значение коэффициента регулятора $k_р^{кр}$, когда система потеряет устойчивость, увеличивая коэффициент регулятора с шагом 10 % от оптимального $k_р^о$.

Результаты однофакторного эксперимента представить в виде семейства АФЧХ (по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$; $T_{\text{И}}^{\text{о}}$, $+T_{\text{И}}$, $-T_{\text{И}}$; $T_{\text{Д}}^{\text{о}}$, $+T_{\text{Д}}$, $-T_{\text{Д}}$).

По ПХ замкнутой САР во временной области по управлению и возмущению найти параметры качества системы: время регулирования $t_{\text{р}}$, максимальное динамическое отклонение σ, перерегулирование η для каждого опыта однофакторного эксперимента, многократно повторяя пункт 4. Выявить закономерности влияния варьируемых параметров регулятора на показатели качества системы. Результаты однофакторного эксперимента представить в виде семейства ПХ (по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$; $T_{\text{И}}^{\text{о}}$, $+T_{\text{И}}$, $-T_{\text{И}}$; $T_{\text{Д}}^{\text{о}}$, $+T_{\text{Д}}$, $-T_{\text{Д}}$ отдельно для системы по управлению и возмущению).

6. Результаты синтеза и анализа САР представить следующим образом:

А. Исходные данные ОУ (передаточная функция), полученные базовые (оптимальные) настройки для систем с П-регулятором (коэффициент передачи регулятора $k_{\text{р}}$), ПИ-регулятором (коэффициент передачи регулятора $k_{\text{р}}$, постоянная интегрирования $T_{\text{И}}$) и ПИД-регулятором (коэффициент передачи регулятора $k_{\text{р}}$, постоянная интегрирования $T_{\text{И}}$, постоянная дифференцирования $T_{\text{Д}}$), а также базовые выходные параметры переходных процессов САР (время регулирования $t_{\text{р}}$, максимальное динамическое отклонение σ, перерегулирование η).

Б. Графики базовой АФЧХ разомкнутой САР и базового переходного процесса замкнутой САР, семейство графиков (фрагменты первого перехода АФЧХ через 180° и переходных процессов), полученных при проведении однофакторного эксперимента, при которых САР остается устойчивой при детерминированном изменении параметров регуляторов. Результаты однофакторного эксперимента представить в виде семейства АФЧХ, ПХ отдельно для системы по управлению и возмущению (по 3 графика при $k_{\text{р}}^{\text{о}}$, $+\Delta k_{\text{р}}$, $-\Delta k_{\text{р}}$; $T_{\text{И}}^{\text{о}}$, $+T_{\text{И}}$, $-T_{\text{И}}$; $T_{\text{Д}}^{\text{о}}$, $+T_{\text{Д}}$, $-T_{\text{Д}}$). В случае расходящегося переходного процесса (неустойчивой САР) графики ПХ представить отдельно.

В. Выявить и написать выводы о закономерности влияния варьируемых параметров регулятора на устойчивость и показатели качества системы.

2.6. СТОХАСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ

2.6.1. Имитационное моделирование

Имитационная модель реального процесса (объекта, явления) – программа для ЭВМ, реализующая упрощенную модель этого процесса вместе с алгоритмом, описывающим течение этого процесса [2.10]. Когда компьютер выполняет эту программу, он «имитирует» течение реального процесса. Меняя различные параметры программы, можно имитировать течение реального

процесса в различных условиях. Таким образом, возникает возможность осуществления следующего диалога:

Вопрос: (его задает исследователь, лицо, принимающее решение (ЛПР), и т.п.): «Что произойдет с объектом (процессом, явлением), если…?».

Ответ: (его дает ЭВМ, «проигрывая» заложенную в нее имитационную модель): «В заданных условиях с объектом (процессом, явлением) произойдет следующее…».

Организованный таким образом диалог человека с компьютером позволяет проводить те или иные эксперименты (они называются имитационными, вычислительными или машинными), получая при этом информацию, которая может быть отнесена к реальному процессу (объекту, явлению). Разумеется, полученная информация будет иметь тем большее отношение к реальности, чем более удачно построена имитационная модель. Схема имитационных исследований представлена на рис. 2.68.

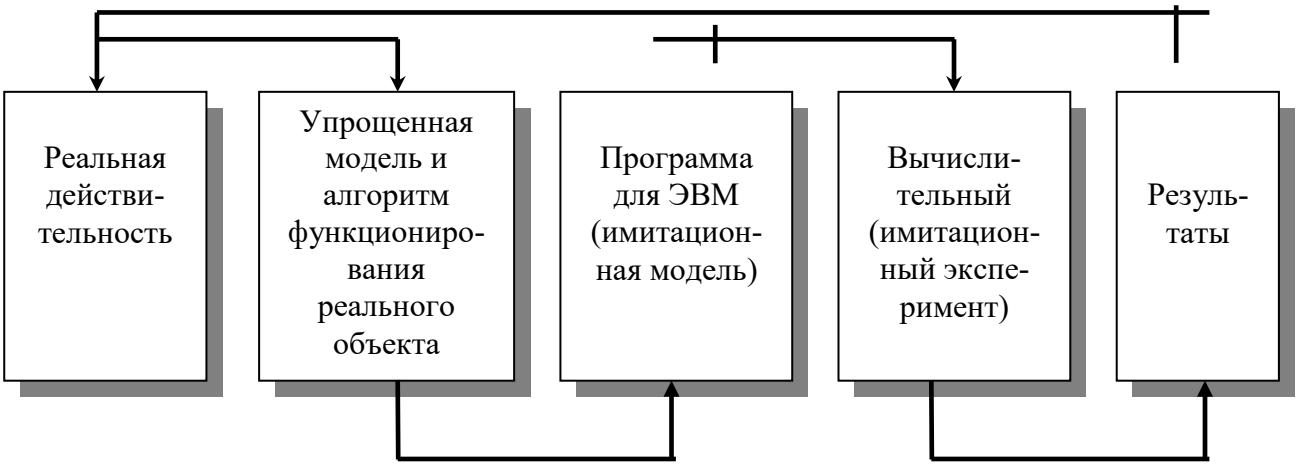


Рис. 2.68. Схема имитационных исследований

Термин «имитационное моделирование» возник как перевод английского выражения *«simulation modeling».* Если обратиться к русским эквивалентам этих слов, то станет ясно, что перевод был сделан не совсем удачно, так как в нем явно имеется тавтология. Однако, поскольку термин уже получил широкое распространение, менять его едва ли возможно и целесообразно. Важнее хорошо понимать, что за ним стоит. Когда говорят «имитационная модель», обычно имеют в виду, что ей свойственны большая, чем в обычных математических моделях, близость к реальному объекту, возможность воспроизводить широкий спектр свойств объекта, использование эмпирического материала, кроме того, всегда сохраняется возможность дальнейшего уточнения модели и др. Говоря об имитационном моделировании, в него включают как процесс создания модели, так и ее исследование (проведение вычислительных экспериментов) с помощью ЭВМ.

Р. Шеннон [2.11] дает следующее определение: «…имитационное моделирование есть процесс конструирования реальной системы и постановки экспериментов на этой модели с целью оценить (в рамках ограничений,

накладываемых некоторым критерием или совокупностью критериев) различные стратегии, обеспечивающие функционирование данной системы».

Как видно из этого определения, имитационное моделирование не очень приспособлено для выяснения причин того или иного явления (оно не дает ответа на вопрос «Почему?»), его роль гораздо более прагматична – давать ответ на вопрос: «Что будет, если…?».

2.6.1.1. Основные этапы имитационного моделирования

Процесс имитационного моделирования (создание модели и алгоритма функционирования реального объекта, создание программы и проведение вычислительных экспериментов) является итеративным. При этом системный анализ чередуется и сочетается с составлением модели и алгоритма, вычислительными экспериментами, корректировкой модели и т.п.

Имитационное моделирование состоит из следующих взаимозависимых и, зачастую, пересекающихся во времени этапов [2.10]:

1) постановка проблемы, формулировка цели моделирования;
2) системный анализ моделируемого объекта; построение концептуальной модели;
3) составление модели и алгоритма функционирования реального объекта, при необходимости – структуризация модели;
4) программная реализация и вычислительные эксперименты;
5) анализ результатов экспериментов и коррекция модели, алгоритма, программы;
6) проведение имитационных экспериментов в целях решения поставленной проблемы.

Охарактеризуем каждый этап.

1. Имитационное моделирование, как и всякое другое, должно начинаться с постановки проблемы, с ясного и четкого понимания целей моделирования. Желательно перечислить те вопросы, ответы на которые должны быть получены в ходе вычислительных экспериментов.

Необходимость построения имитационной модели, как правило, возникает в процессе прикладных исследований конкретного объекта. Поэтому одной из важнейших особенностей этого этапа моделирования является непременное участие в разработках человека (или группы людей), который будет использовать будущую модель для решения поставленной проблемы, т.е. ЛПР. Нужно отметить, что в реальной жизни цели редко бывают четко очерченными. Это существенно затрудняет точную постановку проблемы. Поэтому упомянутый выше список вопросов следует обязательно согласовывать с ЛПР. С ним же следует решать вопрос о масштабах задачи: о ее объеме, границах в

пространстве, продолжительности моделируемого отрезка времени (интересуют ЛПР долгосрочные или краткосрочные эффекты) и т. д.

Необходимо уточнить объект исследования: систем в целом или отдельные ее части. Лишь после этого целесообразно переходить к последующим этапам.

2. С учетом целей моделирования следует выявить существенные особенности изучаемого объекта, осуществить отбор необходимых сведений о нем и построить так называемую концептуальную модель объекта. Эта модель представляет формализованное описание объекта (словесное описание, представление в виде схем, диаграмм и т. п.), являющееся основой для создания имитационной модели.

Построение концептуальной модели предполагает:

а) выявление основных процессов, которые должны быть учтены при моделировании;

б) выявление тех основных характеристик объекта, которые необходимо иметь для решения исходной проблемы;

в) определение множества переменных и параметров, которые влияют на динамику этих характеристик;

г) определение множества входных и выходных данных модели;

д) установление границ и законов взаимодействия объекта с окружающей средой (в частности, определение законов случайных воздействий на объект);

е) разработку причинно-следственных связей, временных отношений и гипотез, согласно которым осуществляется взаимоувязка всех перечисленных компонент в единую систему – имитационную модель.

3. Далее осуществляется переход от качественных зависимостей концептуальной модели к точному алгоритмическому описанию. Исходным пунктом при этом является задание вектора состояния модели, компоненты которого – это те характеристики изучаемого объекта, которые выделены при построении концептуальной модели как базовые, несущие в себе необходимую информацию для решения поставленной проблемы

$$X^t = (X_1^t, X_2^t, \ldots, X_n^t).$$

Часто бывает полезным сразу выделять «управляемые компоненты» вектора состояния. В общем случае каждая компонента вектора X^t есть функция времени, зависящая также от множества значений параметра $\{A\}$, некоторого подмножества компонент вектора состояния X^t, множества внешних факторов η^t и управляющих воздействий U^t, т. е.

$$X^t = X^t(X^t, \eta^t, A, U^t, t).$$

В зависимости от цели моделирования (целевой функции), от проблемы, которую необходимо разрешить с помощью модели, вводится так называемое системное время, моделирующее ход времени в реальной системе. Различают

два типа шкал модельного времени: равномерный и событийный. Для первого типа характерно введение некоторого постоянного шага Δt изменения времени. В этом случае вектор состояния модели рассматривается в моменты $t + k\Delta t$, где k – целое положительное число (разумеется, величина $t + k\Delta t$ не должна превосходить T – величины моделируемого промежутка времени). В этом случае каждому моменту реального времени, t_i^p ставится в соответствие момент модельного времени t_i^m, причем

$$t_1^m - t_2^m = A(t_1^p - t_2^p),$$

где A – масштабный коэффициент пропорциональности.

Если моделируемый объект изменяется лишь при наступлении некоторого события, а в остальные моменты времени остается без изменений, пользоваться равномерной шкалой неудобно, поскольку вектор состояния на отрезке времени между двумя событиями остается постоянным. В этом случае отсчет времени в модели ведется «по событиям», т. е. каждый последующий момент времени в модели наступает только тогда, когда в ней моделируется наступление некоторого события.

После, а иногда и параллельно с заданием вектора состояния системы и выбором временного шага производится декомпозиция модели и выявление ее блочной конструкции (если в этом есть необходимость). С этой целью множество переменных модели делится на непересекающиеся подмножества, в каждое из которых входит группа «однородных» переменных. Понятие «однородности», как правило, определяется самим исследователем в зависимости от цели моделирования, количества и качества исходных данных. Чаще всего однородными, входящими в один блок, считаются переменные, описывающие отдельный процесс, какую-либо подсистему или элемент исходного объекта, группу факторов, имеющих одну и ту же природу (например, климатообусловленные или антропогенные факторы и т. п.). В результате декомпозиции модель представляется в виде комплекса взаимосвязанных подмоделей – блоков, которые взаимодействуют по определенным законам и в итоге позволяют провести имитационное исследование объекта.

Блочный принцип построения модели имеет целый ряд преимуществ, особенно ощутимых при создании сложных имитационных моделей. Прежде всего, если модель достаточно сложна, требует значительного объема памяти и машинного времени, то возможности использования такой модели для имитационного эксперимента оказываются весьма ограниченными. Единственным выходом в этом случае оказывается декомпозиция модели для того, чтобы программы, реализующие отдельные блоки, работали последовательно и обменивались информацией по тем или иным правилам.

Поскольку блоки описывают различные подсистемы, процессы, факторы, системное время для каждого из них может быть различным. Оно определяется, исходя из внутренних потребностей блока. Если, к примеру, в блоке

описываются «быстрые» изменения, то время исчисляется мелкими единицами измерения: секундами, часами, сутками (скажем, при моделировании процесса фильтрации воды в почве после дождя, процесса накопления автомобилей перед светофором и т. п.); если изменения объекта во времени происходят достаточно медленно (рост дерева, старение оборудования, глобальные изменения климата земли и т. п.), то время исчисляется месяцами, годами, десятилетиями и даже тысячелетиями. Однако необходимо позаботиться о том, чтобы в итоге все результаты по отдельным блокам были сведены к единому системному времени, принятому для модели в целом.

После завершения декомпозиции модели следует приступить к разработке отдельных ее блоков. Для каждого из них:

а) уточняются и конкретизируются те гипотезы, которые непосредственно относятся к процессам, аспектам, элементам, «принадлежащим» данному блоку;

б) определяется соответствующее подмножество входных и выходных данных, причем эти данные могут принадлежать как ко множеству «входов» и «выходов» общей модели, так и ко множеству локальных (или внутренних) входных и выходных данных других блоков;

в) формируется множество параметров;

г) формализуются основные законы взаимодействия элементов блока.

При этом происходит переход от качественных зависимостей концептуальной модели к точным количественным зависимостям и логическим схемам взаимодействия элементов внутри блока.

Что касается последнего, то в практике имитационного моделирования далеко не все связи концептуальной модели удается отразить в виде зависимостей теоретического характера. Часто приходится вводить эмпирические зависимости, полученные на основе данных натурных наблюдений, в результате обобщения опыта моделирования подобных объектов и т. п. Неоценимую роль на этапе формализации играет экспертная оценка полученных эмпирических зависимостей, коэффициентов и т. п.

Существенное влияние на качество разработки отдельных блоков оказывает выбор различных математических средств моделирования: аппарата дифференциальных уравнений, статистического моделирования, методов теории оптимального управления, алгоритмизации логических конструкций и т. п. Важным также является то, в каком сочетании эти средства используются для решения исходной проблемы, как согласуется выбор методов с возможностями ЭВМ. Поэтому на этапе разработки блоков необходимо участие не только математиков и экспертов, но и системного программиста, знакомого с характеристиками вычислительной техники, математическим и техническим обеспечением процесса счета. Это позволит оценить возможности программы, реализующей на ЭВМ создаваемую модель, определить пути возможно нежелательных, но часто так необходимых упрощений.

4. Заключительным в построении модели является объединение блоков в имитационную модель на базе стандартного или специально созданного математического обеспечения. Здесь особую роль играет выбор языка программирования: либо это будут универсальные языки типа Си, Паскаль, Фортран и т. п., либо специализированные языки имитационного моделирования типа ДИНАМО, GPSS, Симула и т. д. Проблема носит принципиальный характер, так как специализированные языки удобны для программирования, отличаются концептуальной направленностью, что позволяет при составлении несложных имитационных моделей опустить этап построения формальной модели. Однако такие языки требуют специальных трансляторов, которые не всегда входят в стандартное математическое обеспечение ЭВМ. В свою очередь, использование универсальных языков, как правило, сильно увеличивает объем программ, делает их громоздкими и «трудночитаемыми» для пользователя, хотя и допускает проведение имитационных экспериментов практически на любых ЭВМ, что расширяет область практического применения модели.

Составлению машинных программ, реализующих всю модель в целом, предшествуют испытания и отработка различных схем взаимодействия блоков. Здесь удобно бывает рассматривать имитационную модель как коллектив автоматов с памятью и без нее, детерминированных или стохастических, а работу модели – как изучение с помощью ЭВМ коллективного поведения автоматов в случайной или детерминированной среде.

На этом заканчиваются первые четыре этапа построения имитационной модели. Они представлены на рис. 2.69.

5. Маловероятно, что построенная имитационная модель сразу же окажется удачной. Скорее всего, в ней обнаружатся ошибки. Для отыскивания этих ошибок и используются вычислительные эксперименты как с отдельными блоками, так и с системой в целом. Причем желательно проведение таких экспериментов, результат которых по тем или иным причинам может быть предсказан. Тогда отклонение результата от прогнозируемого служит индикатором наличия ошибок либо в модели, либо в алгоритме, либо в программе, либо сразу в нескольких местах.

Устранение ошибок продолжается до тех пор, пока результаты экспериментов придут в соответствие с прогнозируемыми. После этого можно считать, что имитационная модель создана.

Однако имитационное моделирование состоит не только в построении самой имитационной модели, т. е. инструмента для проведения научных исследований, но и в применении этого инструмента. Необходимо уметь использовать эту модель в исследовательских целях – тех целях, для которых она создавалась.

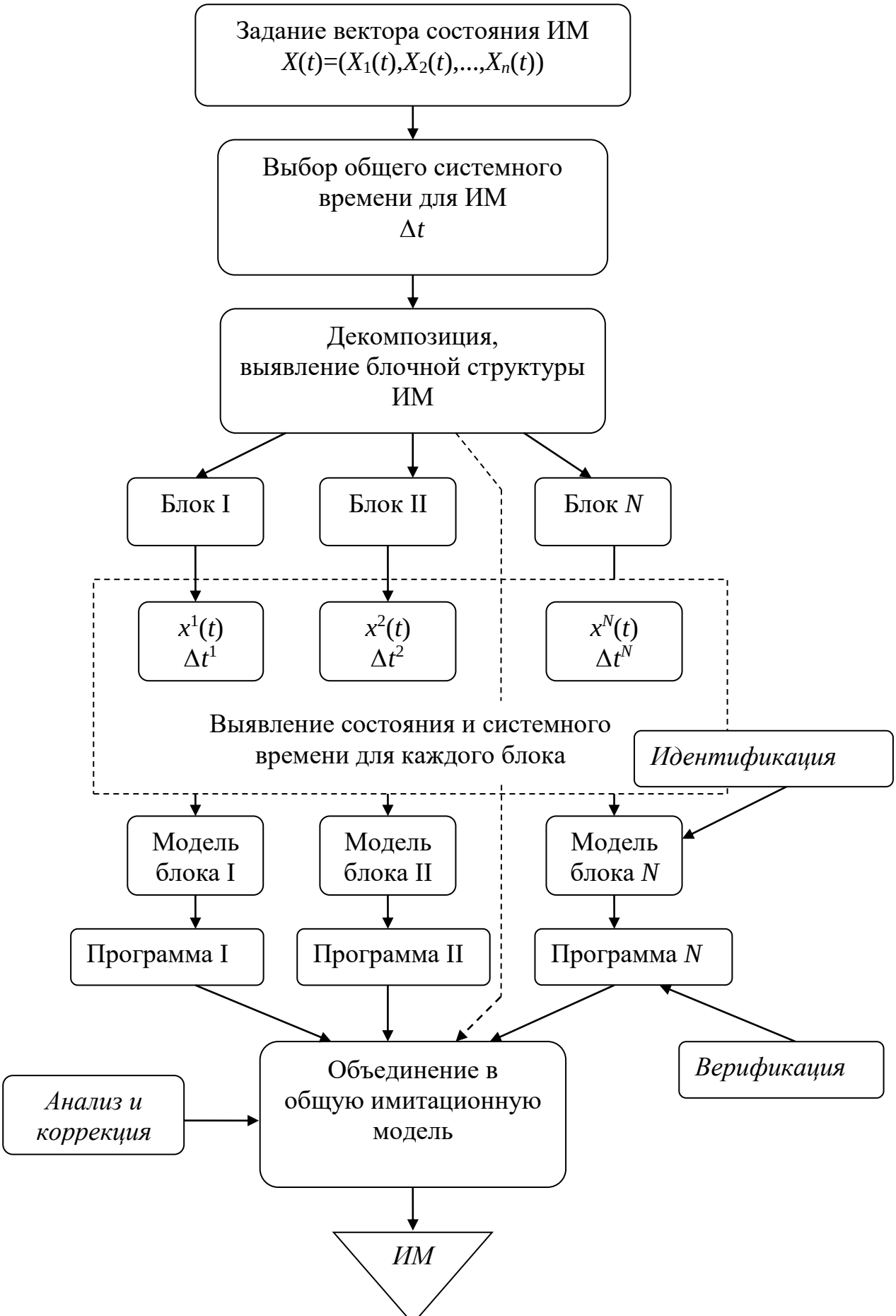


Рис. 2.69. Первые четыре этапа построения имитационной модели

2.6.1.2. Идентификация и верификация имитационной модели

Итак, имитационная модель построена. Однако прежде чем ее использовать, необходимо решить следующие задачи:

1. Выбрать числовые значения неопределенных пока числовых параметров (идентификация).

2. Убедиться, что при этих значениях параметров модель хорошо соответствует моделируемому объекту, адекватна ему (верификация, проверка адекватности).

Приведем наиболее распространенную постановку задачи идентификации. Пусть $\vec{y} = \vec{y}(t, \vec{\alpha})$ – вектор выходных характеристик имитационной модели, зависящей от набора параметров $\vec{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_m\}$. Предположим, что имеется ряд известных векторов $\vec{y}(t)$, $t \in [t_1, t_2]$, представляющих собой результаты натурных наблюдений за моделируемым объектом. Разобьем промежуток $[t_1, t_2]$ на две части: $[t_1, \tau]$ – обучающий промежуток и $[\tau, t_2]$ – экзаменующий промежуток.

Задача идентификации состоит в том, чтобы найти такой набор параметров $\vec{\alpha}$, который доставляет минимум функционалу

$$\sum_{t=t_1}^{\tau} (\vec{y}(t) - \vec{y}(t, \vec{\alpha}))^2 .$$

Для решения этой задачи могут быть использованы известные численные методы нахождения экстремума функции многих переменных.

Обозначим найденный в результате решения этой задачи наилучший набор параметров через $\vec{\alpha}*$. Используя в модели именно эти параметры, получим выходные характеристики в виде $\vec{y} = \vec{y}(t, \vec{\alpha}*)$. На этом решение задачи идентификации закончено.

Перейдем к задаче верификации, которая, однако, не столь формализована. Прежде всего убедимся, что на экзаменующем промежутке времени $[\tau, t_2]$ расчетная траектория $\vec{y} = \vec{y}(t, \vec{\alpha}*)$ близка к фактической.

Сравнение этих двух траекторий позволяет судить об адекватности модели. Существуют специальные методы оценки близости траекторий. Один из них состоит в вычислении коэффициента несовпадения

$$U = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T} (\vec{y}(t) - \vec{y}*(t))^2}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T} \vec{y}^2(t)} + \sqrt{\frac{1}{T}\sum_{t=1}^{T} (\vec{y}*)^2(t)}}, \quad 0 \le U \le 1.$$

Чем ближе U к нулю, тем ближе модельная траектория к фактической. В случае, когда U равен единице, модель не является адекватной и требует либо

перестройки структуры, замены или уточнения гипотез, либо идентификации по более полным и достоверным данным.

Однако даже если близость траекторий имеет место, еще нет гарантии того, что модель адекватна реальному объекту. Естественно потребовать, чтобы выполнялись следующие условия.

1. Машинная реализация соответствует формальной модели.

2. «Динамика» модели соответствует «динамике» реального объекта.

3. Результаты моделирования правильно интерпретируются.

Проверка адекватности в этом смысле осуществляется на основе экспертного анализа и статистических методов.

Под соответствием машинной и формальной моделей понимается, во-первых, идентичность их алгоритмических структур (сохраняется ли логика построения модели при машинной реализации?) и, во-вторых, совпадение областей варьирования компонент вектора состояния формальной и машинной моделей. Второе требование означает, что численные методы, которые используются для реализации модели на ЭВМ, не должны давать такую погрешность, которая выводит некоторую компоненту из области допустимых значений. Например, если шаг интегрирования системы обыкновенных дифференциальных уравнений с положительно-определенными переменными выбран слишком большим, то можно получить отрицательные значения искомых переменных, что не соответствует их смыслу.

Соответствие имитируемой и реальной динамики частично проверяется уже на стадии верификации модели. Однако известно, что практически любую имитационную модель можно отладить до нужной степени совпадения моделируемой и фактической траекторий. На рис. 2.70 приведен пример прогнозирования по модели, которая хорошо аппроксимирует фактическую траекторию на ретроспективном периоде, но дает абсолютно неверный прогноз, т. е. неадекватно отражает поведение реального объекта.

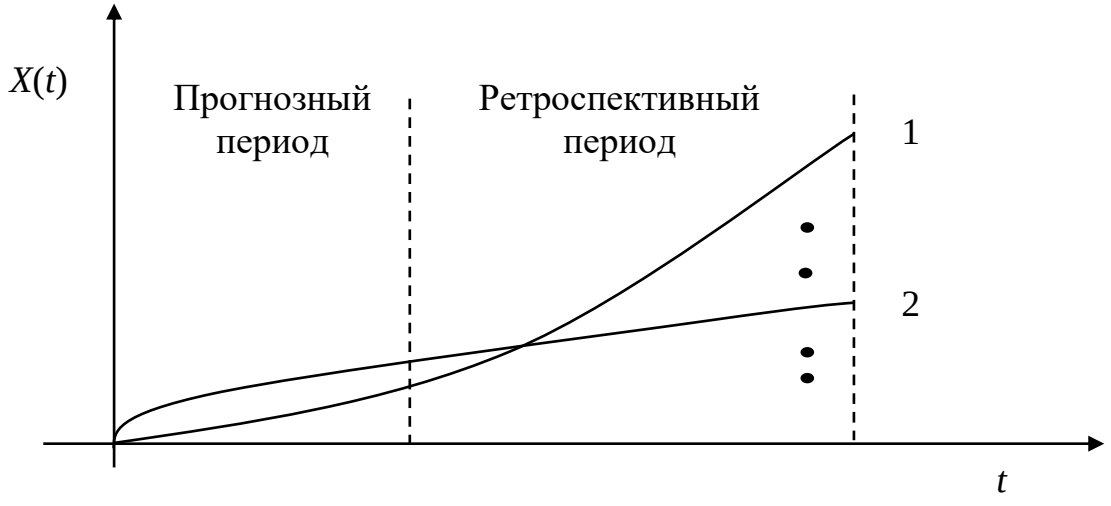


Рис. 2.70. Прогнозирование по модели:
1 – фактическая траектория; 2 – расчетная траектория

Следовательно, для того чтобы оценить адекватность модели, необходимо качественно исследовать ее динамические свойства путем проведения серии тестовых расчетов. Можно выделить два основных типа тестов. К первому относятся тесты, в которых задаются правдоподобные значения входов и управляющих воздействий. Если в этом случае расчеты по модели не противоречат известным законам поведения реального объекта (нашим представлениям о нем), то это говорит в пользу адекватности модели. В противном случае необходимо найти причины несогласованности и перестроить модель. Второй тип тестов основан на использовании критических ситуации, т. е. данных, которые не характерны для исследуемого объекта, но тем не менее могут иметь место. Эти тесты особенно важны для сложных моделей, предназначенных для долгосрочного прогнозирования, поскольку чем лучше модель описывает поведение объекта в критических условиях, тем больше можно быть уверенным в правильности расчетов для нормальных условий.

Адекватность модели характеризуется также ее чувствительностью по отношению к изменениям параметров и начальных значений вектора состояния. Если результаты исследований с помощью модели существенно изменяются при малых возмущениях параметров и незначительных отклонениях от начальных данных, другими словами, если модель не является устойчивой, то ее нельзя считать адекватной (при условии, конечно, что исследуемый объект обладает устойчивостью в этом смысле). Требование устойчивости тем более важно, чем менее точно могут быть определены параметры модели.

Анализируя адекватность модели, следует помнить о цели исследования, о проблеме, для решения которой разрабатывается модель. Дело в том, что конкретная модель может вполне отвечать одной цели и быть совершенно непригодной для решения других задач. Здесь важен вопрос об области применимости модели, Модель пригодна только тогда, когда она дает возможность реализовать цели исследования.

Основными методами оценки адекватности и пригодности модели являются, безусловно, неформальные (метод экспертного оценивания, соблюдение принципа «здравого смысла» и т. п.). Однако в некоторых случаях удается формализовать эту процедуру. В частности, весьма полезными оказываются различные статистические методы, спектральный анализ и др.

Итак, модель идентифицирована, верифицирована, проверена на адекватность. Если все этапы выполнены успешно, то она является готовым инструментом исследования поставленной проблемы и можно переходить к главному этапу имитационного моделирования – проведению имитационного эксперимента, которое сопровождается, с одной стороны, планированием эксперимента, а с другой – обработкой результатов этого эксперимента.

Эксперименты делятся по цели исследования на два основных типа: дескрипторные и оптимизационные. Эксперименты первого типа проводятся в

целях исследования объекта. Другой тип включает эксперименты, направленные на выявление наилучших стратегий управления исходным объектом.

Результаты имитационных экспериментов должны быть обработаны специальными методами и представлены пользователю в удобном для него виде.

2.6.1.3. Планирование имитационных экспериментов

Проведение имитационного эксперимента – главный этап имитационного исследования, включающий, кроме расчетов по модели, планирование эксперимента и обработку результатов эксперимента. Сразу же возникает вопрос о том, при каких внешних воздействиях проводить расчеты, сколько расчетов проводить для того, чтобы быть уверенным в верности полученного решения и т. д. Все эти проблемы могут быть решены в процессе планирования эксперимента.

Цель эксперимента – установить связь между воздействиями на модель и ее откликом на это воздействие. На этом этапе модель можно представить в виде

$$y = f(x),$$

где x – воздействие на модель или фактор; y – результат воздействия или реакция; f – поверхность реакции.

В общем случае x и y есть вектор-функции, зависящие от времени. Любой имитационный эксперимент в этом случае может быть направлен либо на исследование поверхности реакции (задачи прогнозирования, например), либо на поиск максимума или минимума поверхности реакции в некотором пространстве факторов (задачи оптимального управления объектом и т. п.).

Факторы могут быть либо количественными, либо качественными. Рассмотрим только факторы первого типа.

Пусть для простоты поверхность реакции описывается функцией, зависящей от двух количественных факторов:

$$f = f(x_1, x_2),$$

причем x_1 и x_2 могут принимать дискретные фиксированные значения из областей X^1 и X^2. Значения фактора назовем уровнем, а совокупность всех возможных пар (x_1, x_2) – полным факторным планом (см. рис. 2.71).

Чем больше точек плана будет рассмотрено, т. е. чем полнее построен план, тем точнее представления о виде поверхности реакции. Однако, несмотря на это, применение полных факторных планов ограничено и возможно лишь в случае незначительного числа факторов и их уровней. Более часто используются неполные факторные планы, требующие меньшего числа точек плана и не приводящие при этом к ощутимым потерям информации о поверхности реакции. Здесь в основном исследуются несколько главных

факторов, а неполные факторные планы применяются для «отсеивания» несущественных факторов. Процедура отсеивания состоит в последовательном построении неполных планов.

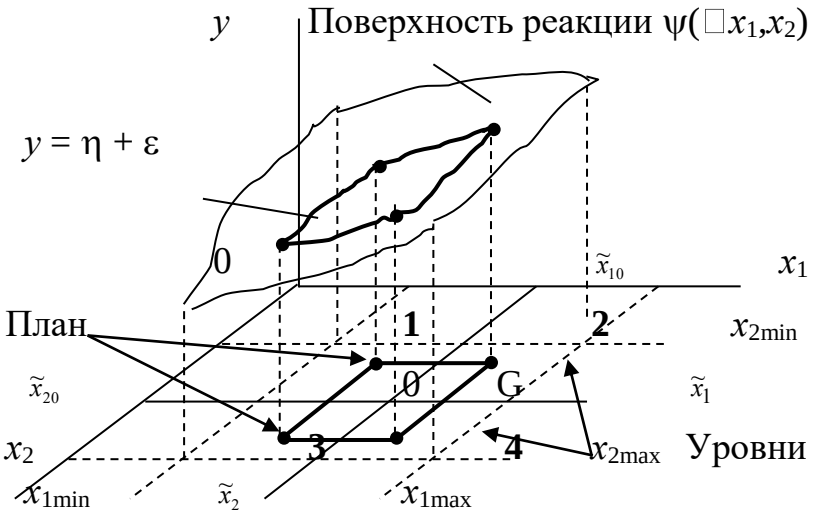


Рис. 2.71. Полный факторный план

Вернемся к примеру двухфакторной модели

$$f = f(x_1, x_2),$$

где факторы x_1, x_2 имеют более 3 уровней каждый. Сначала имитационный эксперимент проводится для начальных значений этих факторов (x_1^0, x_2^0), а затем задается некоторое фиксированное изменение каждого уровня: δ_1 и δ_2, где $\delta_1 = \frac{x_1}{n_1}$, $\delta_2 = \frac{x_2}{n_2}$, n_1, n_2 – число уровней x_1 и x_2 соответственно, и строятся неполные факторные планы, где x_1, x_2 – средние уровни факторов x_1 и x_2. Реализация этих планов дает четыре точки поверхности реакции $f^1_{+\delta 1}$, $f^1_{-\delta 1}$, $f^2_{+\delta 2}$, $f^2_{-\delta 2}$, по которым можно судить о степени влияния каждого фактора. Так, если

$$| f^1_{+\delta 1} - f^1_{-\delta 1} | > | f^2_{+\delta 2} - f^2_{-\delta 2} |,$$

то x_1 является более существенным фактором, и наоборот.

Еще одна типичная задача планирования эксперимента состоит в аппроксимации истинной поверхности реакции некоторой функцией φ_n, зависящей от тех же факторов. Как правило, удается построить линейную зависимость $\varphi_n = \sum_{i=1}^{n} a_i x_i + a_0$, где a_i – коэффициенты линейного многочлена.

Пусть, например, $n = 2$. Тогда для построения полинома

$$\varphi_2 = a_1 x_1 + a_2 x_2 + a_0$$

требуется проведение имитационного эксперимента по полному двухфакторному плану с уровнями $x_{1,2}^0 \pm \delta_{1,2}$.

В случае неудовлетворительной аппроксимации есть возможность строить полиномы более высокой степени. Например, полином второй степени

$$\varphi = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_{11} x_1^2 + \alpha_{22} x_2^2 + \alpha_{12} x_1 x_2.$$

Его коэффициенты находятся за счет дополнительных изменений и на основе новых экспериментов. Процесс продолжается до тех пор, пока аппроксимация не даст удовлетворительных результатов.

Описанный метод называется методом поверхности реакции.

Более узкой является проблема поиска экстремумов поверхности реакции, для решения которой используются известные методы оптимизации на заданном множестве значений факторов. Среди них отметим метод наискорейшего спуска (или подъема), который состоит в исследовании поверхности реакции в окрестности некоторой точки с помощью линейных аппроксимирующих поверхностей-гиперплоскостей. Такие гиперплоскости обычно строятся с помощью простых экспериментов, как правило, однофакторных (рис. 2.72).

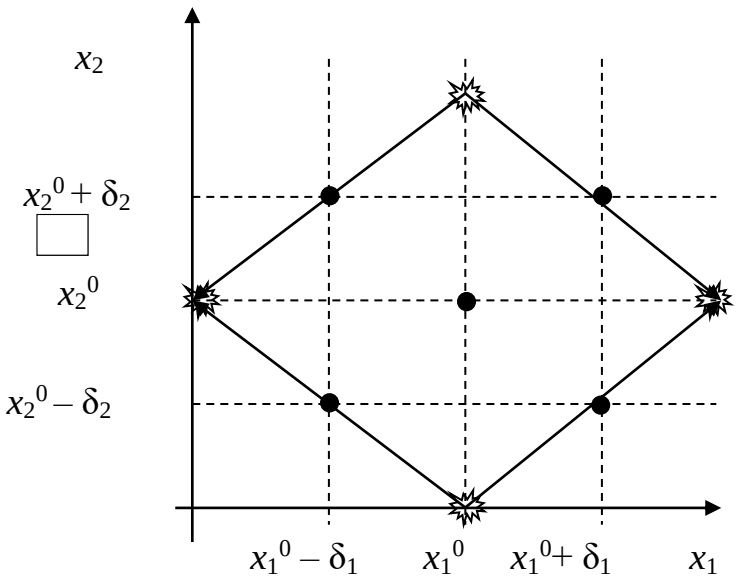


Рис. 2.72. Однофакторный эксперимент:
• – начальные изменения; * – дополнительные изменения

По построенной гиперплоскости определяется направление движения к точке оптимума, а затем в этом направлении делается небольшой «шаг». Далее процедура повторяется.

Метод наискорейшего спуска не гарантирует минимума. Если допустить, что поверхность реакции имеет несколько локальных минимумов, то целесообразно несколько раз применять этот метод, отправляясь, всякий раз от различных, сильно отличающихся начальных условий.

Заметим, что линейная аппроксимация вблизи точки оптимума оказывается неэффективной. В окрестности точки оптимума, где поверхность реакции почти стационарна, используется аппроксимация более высокого порядка – например квадратичным полиномом.

Все перечисленные методы планирования эксперимента относятся к детерминированным моделям. Для стохастических моделей однократная реализация построенного плана не позволяет получить желаемую информацию об изученной поверхности реакции. В этом случае необходимо несколько раз реализовывать один и тот же план с различными начальными состояниями генератора случайных чисел (каждая реализация называется репликой). Определение объема выборки (количества реплик) в имитационном эксперименте представляет собой очень трудную, но важную задачу. С одной стороны, увеличение объема ведет к увеличению затрат машинного времени и, тем самым, денежных средств. С другой стороны, чем больше количество реплик, тем более достоверна информация, полученная с помощью модели, и меньше возможные потери, обусловленные использованием недостоверной информации. Минимизация суммарных потерь всякий раз осуществляется с помощью методов статистического анализа и методов оптимизации.

2.6.2. Классификация методов теории вероятностей и математической статистики

Вероятностные (стохастические) модели описывают ситуации, в которых аналогичные причины приводят к различным следствиям, т.е. имеет место элемент случайности. Для построения вероятностной модели необходимо знать, какие величины можно считать случайными, а какие – неслучайными; какой характер имеют законы распределения случайных величии и т. д.

Вероятностные модели можно разделить на две большие группы (см. рис. 2.73):

- математическая модель, в которой можно точно указать законы распределения случайных величин, является *теоретико-вероятностной;*
- математическая модель, в которой заранее нельзя указать законы распределения случайных величин, является *статистической.*

По степени сложности вероятностные модели делятся на три уровня. Простейшие теоретико-вероятностные модели первого уровня – *случайное событие* (СС) и *случайная величина* (СВ), являющиеся соответственно качественной и количественной характеристиками проведенного испытания.

СС может быть простейшим (элементарным) или сложным (выраженным через элементарные). Для описания вероятностных свойств простейшего СС *A* используются стандартные формулы классической (в «схеме случаев») или геометрической вероятности:

$$P(A) = \frac{m}{n},$$

где m – число случаев из пространства элементарных событий, благоприятных событию A; n – общее число случаев, содержащееся в пространстве элементарных событий.

$$P(A) = \frac{\mu(A)}{\mu(D)},$$

где $\mu(A)$ – мера области A; $\mu(D)$ – мера области D, при этом $A \subset D$ и предполагается, что вероятность внутри области D распределена равномерно.

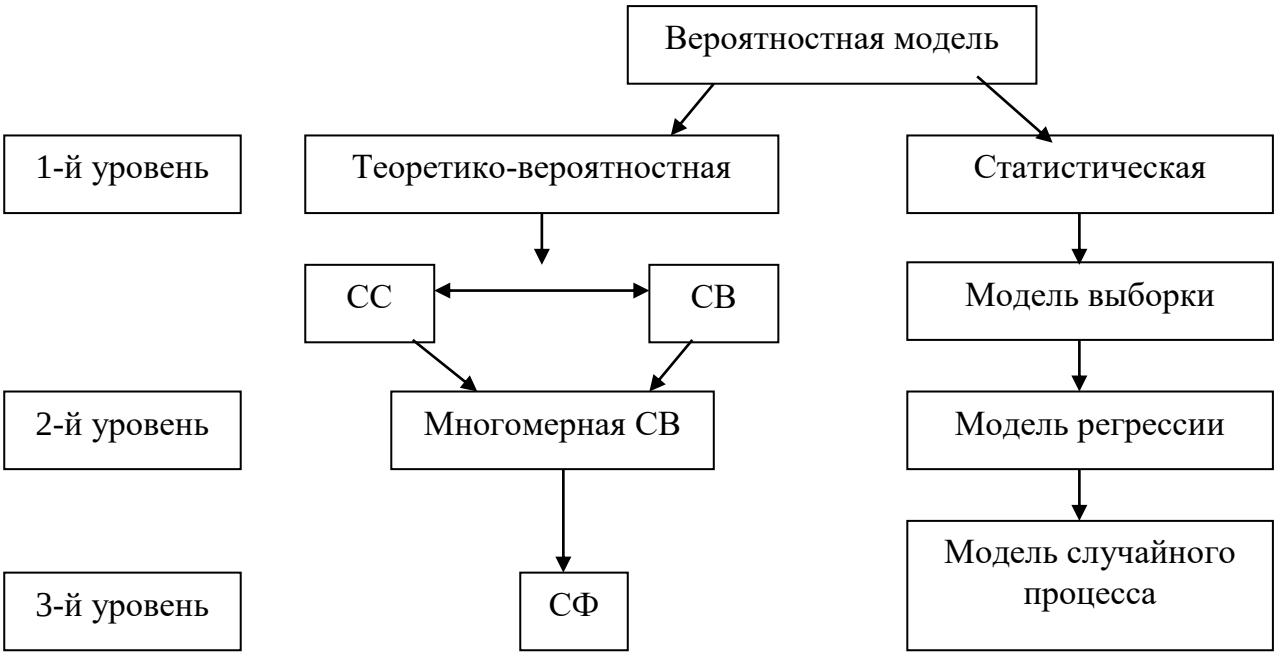


Рис. 2.73. Классификация вероятностных моделей

Для расчета вероятности сложных событий используются теоремы сложения и умножения вероятностей, формула полной вероятности:

$$P(A + B) = P(A) + P(B) - P(A \cdot B);$$

$$P(A \cdot B) = P(A/B) \cdot P(B),$$

где $P(A/B)$ – условная вероятность события A при условии, что событие B произошло;

$$P(A) = \sum_{i=1}^{n} P(H_i) \cdot P(A/H_i),$$

где $P(H_i)$ – вероятность гипотезы H_i (события, при котором может произойти событие A); $P(A/H_i)$ – условная вероятность события A при выполнении гипотезы H_i.

СС и СВ связаны между собой через пространство элементарных событий. При этом вероятностные свойства СВ дискретного типа описываются функцией

дискретного аргумента

$$p_i = f(x_i),$$

где x_i – реализация СВ X; p_i – соответствующая ей вероятность, а вероятностные свойства СВ непрерывного типа описываются функцией $f(x)$ (плотностью распределения), определенной на всей числовой оси, неотрицательной и нормированной там.

Так, например, в биномиальном законе распределения, реализуемом в схеме независимых испытаний, пространство элементарных событий дискретной СВ X есть конечное множество целых чисел, включая 0, т. е.

$$\Omega(X) = \{0, 1, 2, \ldots, n\},$$

а вероятности значений рассчитываются по формуле Бернулли:

$$P_n(X = n) = C_n^m p^m q^{n-m},$$

где $m = \overline{0, n}$; C_n^m – число сочетаний из n по m; p – вероятность появления события в отдельном испытании; $q = 1 - p$.

Нормальный закон распределения – один из основных для непрерывной СВ – X задается плотностью распределения:

$$f(x) = \frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{(x-m_x)^2}{2\sigma_x^2}},$$

определенной для $\forall x \in (-\infty; +\infty)$ и удовлетворяющей условиям $f(x) \geq 0$ и

$$\int_{-\infty}^{+\infty} f(x)dx = 1.$$

Здесь m_x и σ_x – соответственно математическое ожидание и среднее квадратичное отклонение СВ X. Отметим, что эти числовые характеристики являются основными для одномерной СВ X, имеющей любой закон распределения.

Универсальной формой закона распределения, имеющей место как для дискретной, так и для непрерывной СВ, является функция распределения

$$F(x) = P\{X < x\}.$$

Второй уровень теоретико-вероятностной модели, обобщающей модель одномерной СВ, связан с системой случайных величин (многомерной СВ), вероятностные свойства которой не исчерпываются свойствами отдельных величин, образующих систему, а описываются также зависимостью между ними.

Так, для двумерной непрерывной СВ (X, Y) должны рассматриваться частные плотности распределения $f_1(x)$ и $f_2(y)$, а также совместная плотность $f(x, y)$. При этом

$$f_1(x) = \int_{-\infty}^{+\infty} f(x, y)dy;$$

$$f_2(y) = \int_{-\infty}^{+\infty} f(x, y)dx.$$

Однако $f(x, y)$ не всегда может быть определена через $f_1(x)$ и $f_2(y)$. Для n-мерной СВ (X_1, X_2, …, X_n) функции распределения имеют гораздо более громоздкий вид, поэтому на модельном уровне при описании объекта обычно используют набор числовых характеристик.

Для двумерной СВ (X, Y) – это m_x, m_y, σ_x, σ_y, r_{xy}, где r_{xy} – коэффициент корреляции; для n-мерной СВ – это $m_{x_1}, m_{x_2}, ..., m_{x_n}$, а также нормированная корреляционная матрица $||r_{ij}||$, где r_{ij} – коэффициент корреляции между СВ X_i и X_j.

Обобщением модели n-мерной СВ служит модель третьего уровня – случайная функция (СФ) $X(t)$, где t – вещественный параметр.

Графически СФ представима в виде набора реализации $x_i(t)$, где каждая $x_i(t)$ – неслучайная функция ($i = \overline{1, n}$) (рис. 2.74). При фиксированном $t = t_0$ имеем сечение СФ $X(t_0)$, представляющее собой одномерную СВ.

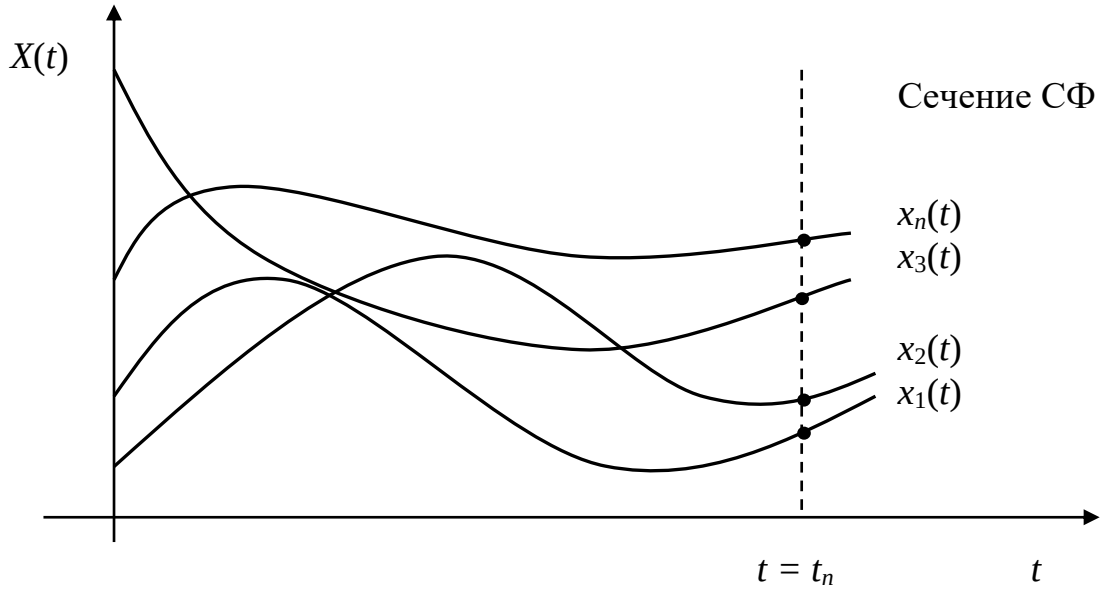


Рис. 2.74. Случайная функция $X(t)$ в виде набора реализации $x_i(t)$

Таким образом, полное вероятностное описание СФ связано с заданием бесконечномерного закона распределения всех ее сечений.

Обычно на модельном уровне ограничиваются рассмотрением СФ в рамках корреляционной теории, т.е. рассмотрением ее математического ожидания $m_x(t)$ и корреляционной функции $K_x(t_1, t_2)$, являющихся неслучайными функциями соответственно одного и двух аргументов.

Важным частным случаем модели СФ является модель стационарной СФ, для которой

$$\begin{cases} m_x(t) = \text{const}; \\ K_x(t_1, t_2) = K_x(\tau), \end{cases}$$

где $\tau = t_1 - t_2$.

Каждому из уровней теоретико-вероятностных моделей соответствует своя статистическая модель.

Поскольку математическая статистика занимается обработкой информации при наличии неопределенности, то в основу построения статистической модели должны быть положены некоторые допущения. В зависимости от их математической сути различают три основные модели математической статистики:

1) модель выборки;

2) модель регрессии;

3) модель случайного процесса.

Каждая из них связана с определенным исходным материалом и решает свои специфические задачи. Однако все они имеют дело с большим объемом информации и достаточно трудоемкими методами ее обработки. Поэтому реализация статистических моделей осуществляется, как правило, на ЭВМ, для которых разработаны стандартные программы определения статистических законов распределения, вычисления статистических характеристик СВ, проверки гипотез по критериям согласия, расчета коэффициентов регрессии, статистических характеристик случайных процессов и т. д.

В некоторых ситуациях получение исходных статистических данных путем специально организованных экспериментов невозможно. В этом случае необходимый статистический материал может быть получен с помощью специально созданных математических моделей. Их основу составляет статистическое моделирование на ЭВМ СС и СВ. Такой метод моделирования называется *методом статистических испытаний*.

Простейшей статистической моделью является одномерная *модель выборки*, в которой предполагается, что исходный статистический материал есть реализация одной СВ X с законом распределения $F(x)$.

Основой для построения модели служит простая случайная выборка, представленная в виде ряда наблюдений (табл. 2.9).

Таблица 2.9

i	1	2	…	n
x_i	x_1	x_2	…	x_n

Например, при измерении расстояния до цели каким-либо прибором за счет чисто случайных причин результат изменяется от опыта к опыту.

Реализация модели связана с построением эмпирических (статистических) законов распределения: $f^*(x)$ – статистическая плотность распределения; $F^*(x)$ – статистическая функция распределения, аналогичная теоретическим законам для СВ X. Графическим представлением $f^*(x)$ является гистограмма, а $F^*(x)$ – кривая накопленных частот.

Найденные законы содержат элемент случайности, т. к. они определены по конечному числу наблюдений. Поэтому для уточнения модели следует провести сглаживание статистического ряда, т. е. выбрать теоретический закон распределения, наилучшим образом описывающий исходный материал. Для этого используются критерии согласия Пирсона, Колмогорова и др.

В некоторых ситуациях ограничиваются получением точечных и интервальных оценок основных числовых характеристик m_x и σ_x^2.

Если результаты наблюдений зависят от некоторого параметра и изменяются от измерения к измерению не только за счет случайных причин, но и за счет существенных, то модель выборки неприменима к данному ряду наблюдений. В этом случае используют *модель регрессии,* которая предполагает, что исходный статистический материал представляет собой реализации СВ, изменяющейся в зависимости от какого-либо параметра (времени, пространственной координаты и т. д.).

Исходный ряд наблюдений имеет вид, приведенный в табл. 2.10, где t_n – значение параметра (i = 1, ..., n); x_n – значение СВ, соответствующее данному значению параметра. Например, изменение температуры воздуха в данной точке в данный момент времени зависит от высоты.

Таблица 2.10

t_i	t_1	t_2	…	t_n
x_i	x_1	x_2	…	x_n

Основным предположением модели регрессии, сводящим ее к рассмотренной ранее модели выборки, является следующее: реализации СВ могут быть представлены в виде суммы

$$x_i = f(t_i) + \delta_i,$$

где $f(t_i)$ – неслучайная функция аргумента t; δ_i – ошибки, составляющие в выборке.

Другими словами, предполагается, что ряд наблюдений (x_1, x_2, …, x_n) является реализацией системы независимых СВ (X_1, X_2, …, X_n), частные распределения которых одинаковы, за исключением математического ожидания, для которого

$$m_{x_i} = f(t_i).$$

Функция $f(t)$ называется сглаживающей, а ее график – линией регрессии. Основной задачей модели регрессии является определение линии регрессии $f(t)$ и оценка точности результата (рис. 2.75).

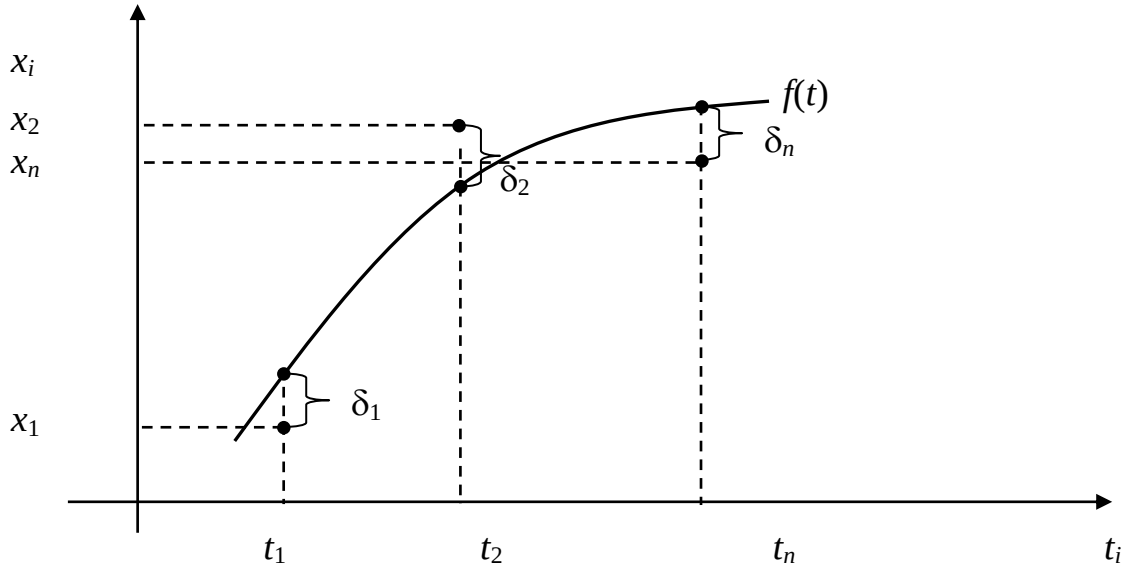


Рис. 2.75. Линия регрессии $f(t)$ и оценка точности результата δ

Сглаживающую функцию обычно задают параметрически: $f(a_1, a_2, \ldots, a_m, t)$, где вид функции известен и вычислению подлежат только параметры $a_1, a_2, \ldots, a_m$ ($m < n$). Для их определения используется стандартная процедура метода наименьших квадратов (МНК), основанная на соотношении:

$$\sum_{i=1}^{n}\left[x_i - f(a_1, a_2, \ldots, a_m, t)\right]^2 = \sum_{i=1}^{n}\delta_i^{\,2} \to \min,$$

т. е. сумма квадратов отклонений результатов наблюдений от сглаживающей кривой должна быть наименьшей.

В большинстве прикладных задач ограничиваются линейной, параболической регрессиями или приводимым к ним.

Модель случайного процесса является обобщением как модели выборки, так и модели регрессии. Исходный ряд наблюдений зависит от параметра так же, как и в модели регрессии, но СВ (X_1, X_2, …, X_n), реализацией которых является данный ряд, зависимы и также зависимы случайные ошибки δ_i в случайных измерениях.

Основное предположение модели случайного процесса состоит в том, что ряд наблюдений, зависящих от параметра, есть значения функции $x(t)$, являющейся реализацией СФ $X(t)$, распределение и характеристики которой заранее неизвестны.

Пусть имеется m рядов наблюдений, проведенных в одних и тех же условиях. Их можно свести в таблицу, в которой в строках расположены

реализации, а в столбцах – сечения по параметру t (табл. 2.11). Данными этой таблицы могут быть, например, многолетние метеорологические наблюдения за температурой воздуха в каком-нибудь пункте в определенный месяц года.

Таблица 2.11

$x(t)$	t			
	t_1	t_2	…	t_n
$x_1(t)$	$x_1(t_1)$	$x_1(t_2)$	…	$x_1(t_n)$
$x_2(t)$	$x_2(t_1)$	$x_2(t_2)$	…	$x_2(t_n)$
…	…	…	…	…
$x_m(t)$	$x_m(t_1)$	$x_m(t_2)$	…	$x_m(t_n)$

Обработка построенной таблицы может вестись как по «столбцам» (модель выборки), так и по «строкам» (модель регрессии). Полученные при различных значениях аргумента t результаты числовых характеристик в дальнейшем аппроксимируются по МНК аналитическими выражениями.

Если по каким-либо причинам получение нескольких статистических рядов невозможно, то для применения модели случайного процесса требуются дополнительные предположения о его характере (например, стационарность процесса).

При *статистическом* моделировании применяется также многомерная *модель выборки* – метод Монте-Карло [2.11]. При этом результаты, полученные при воспроизведении на имитационной модели процесса функционирования системы, являются реализациями случайных величин и функций, и для нахождения характеристик процесса требуется его многократное воспроизведение с последующей статистической обработкой информации.

Основная идея *метода Монте-Карло* заключается в следующем. Пусть необходимо определить функцию распределения случайной величины y. Допустим, что искомая величина y может быть представлена в виде зависимости

$$y = y(\alpha, \beta, \lambda, \ldots \omega),$$

где α, β, λ, … ω – случайные величины с известными функциями распределения.

Для решения задач такого вида применяется следующий алгоритм:

1) по каждой из величин α, β, λ, … ω производится случайное испытание, в результате которого определяется некоторое конкретное значение случайной величины α_i, β_i, λ_i, … ω_i;

2) используя найденные величины, определяется одно частное значение y_i по вышеприведенной зависимости;

3) предыдущие операции повторяются N раз, в результате чего определяется N значений случайной величины y;

4) на основании N значений величины y находится ее эмпирическая функция распределения, математическое ожидание μ и дисперсия σ.

При достаточно большом объеме выборки (числе реализации N) методы оценки распределений и некоторых их моментов следующие [2.4]: математическое ожидание и дисперсия случайной величины ξ, которые соответственно имеют вид

$$\mu_\xi = M[\xi] = \int_{-\infty}^{\infty} x f(x) dx;$$

$$\sigma_\xi^2 = D[\xi] = M[(x-\mu_\xi)^2] = \int_{-\infty}^{\infty} (x-\mu_\xi)^2 f(x) dx,$$

где $f(x)$ – плотность распределения случайной величины ξ, принимающей значения $\xi \le x$.

При проведении имитационного эксперимента со стохастической моделью системы определить эти моменты нельзя, так как плотность распределения, как правило, априори неизвестна. Поэтому при обработке результатов моделирования приходится довольствоваться лишь некоторыми оценками моментов, полученными на конечном числе реализации N. При независимых наблюдениях значений случайной величины ξ в качестве таких оценок используются

$$\bar{x} = \tilde{\mu}_\xi = (1/N)\sum_{i=1}^{N} x_i,$$

$$S_b^2 = \tilde{\sigma}_b^2 = (1/N)\sum_{i=1}^{N} (x_i - \bar{x})^2,$$

где $\bar{x}$ и S_b^2 – выборочное среднее и выборочная дисперсия соответственно. Знак ~ над $\tilde{\mu}_\xi$ и $\tilde{\sigma}_b^2$ означает, что эти выборочные моменты используются в качестве оценок математического ожидания μ_ξ и дисперсии σ_b^2.

2.6.3. Описание работы программного модуля SIANRG

Целью моделирования САР является создание системы, удовлетворяющей определенным требованиям, которые формулируются в виде качественных показателей. Качество функционирования САР определяется по переходной характеристике $h(t)$ – реакции системы на типовое воздействие в виде единичной ступенчатой функции $1(t)$ – или некоторыми параметрами ПФ замкнутой системы.

Для синтеза и анализа одноконтурных САР (см. рис. 2.30) разработан программный модуль *SIANRG*. В этом модуле реализованы различные математические модели ОУ и блоков регулирования (см п. 2.5).

Модель ОУ описывается передаточными функциями (2.4), которые характеризуют реакцию выхода системы на управляющее и возмущающее воздействия. В программном модуле *SIANRG* предусмотрена возможность ввода полиномов числителя и знаменателя в трех формах (2.5) – (2.7'). Упрощенная форма ПФ ОУ (2.8), (2.9). Модель регулятора описывается также передаточными функциями динамических звеньев, соответствующих регулирующим блокам (2.10) – (2.14) или (2.15) – (2.17').

Для составления модели одноконтурной САР проводится синтез, при котором задаются ПФ ОУ, выбирают закон регулирования и определяют настройки блока регулирования (см. п. 2.5.2.5).

Второй этап моделирования – анализ САР позволяет определить устойчивость и качество системы по переходному процессу (см. п. 2.5.2.4). Анализ САР в частотной области позволяет оценить устойчивость замкнутой САР по амплитудно-фазовой частотной характеристике (АФЧХ) разомкнутой САР (см. п. 2.5.2.6). В программном модуле *SIANRG* проверка устойчивости одноконтурной САР осуществляется по критерию Найквиста. Анализ замкнутой САР во временной области позволяет оценить качество устойчивой САР по переходному процессу, вызванному единичным скачком по каналам управления и возмущения (см. п. 2.5.2.7).

Моделирование в программном модуле *SIANRG* начинается с Главного меню (см. п. 2.5.3, рис. 2.36). Меню *Синтез* содержит опции, позволяющие последовательно проводить синтез САР: задание параметров объекта, расчет критической настройки регулятора, расчет оптимальных параметров регуляторов (см. п. 2.5.3, рис. 2.37 – 2.43). Меню *Анализ* (см. рис. 2.44) содержит опции, позволяющие последовательно проводить анализ: ОУ, разомкнутой САР в частотной и временной областях, замкнутой САР по управлению и возмущению (также в частотной и временной областях), разброс параметров объекта, разброс параметров регулятора.

Данный параграф посвящен статистическому анализу по разбросу параметров для ПИД-регулятора (см. рис. 2.76).

Вначале необходимо выбрать один из параметров ПИД-регулятора (коэффициент усиления $k_р$, постоянную интегрирования $T_Д$ или постоянную дифференцирования $T_Д$), задать параметры распределения для выбранной случайной величины (математическое ожидание, дисперсию, число точек расчета) и нажать кнопку *Построить*. На экране появится график нормального распределения выбранного параметра регулятора. Можно сохранить график, используя в данном меню *Файл/Сохранить график*.

Затем зайти в меню *Графики/Распределение* (см. рис. 2.77) и построить распределение исследуемой величины, выбрав параметр распределения регулятора (коэффициент усиления $k_р$, постоянную интегрирования $T_Д$, постоянную дифференцирования $T_Д$), тип анализа САР (по управлению или возмущению), и тип графика – характеристику переходного процесса, как

показателя качества устойчивой САР (время регулирования $t_р$, максимальные динамические отклонения σ, перерегулирование η).

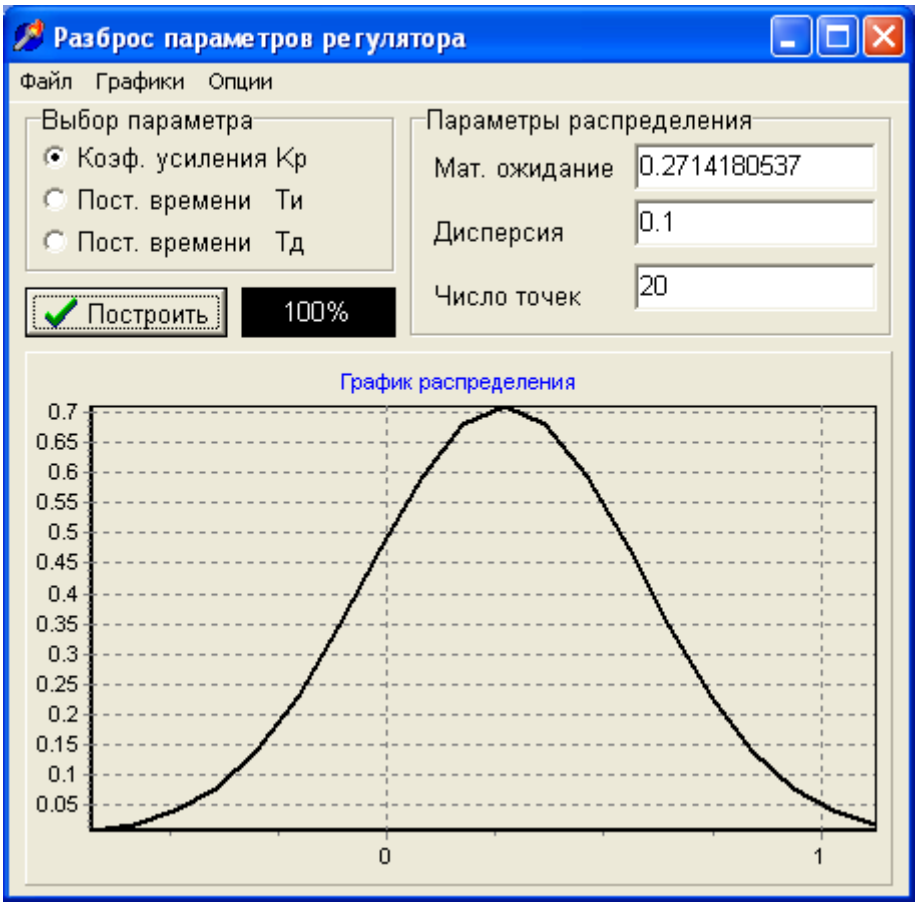


Рис. 2.76. Задание параметров для анализа по разбросу параметров регулятора

Обязательно сразу сохраните график, используя в данном меню *Файл/Сохранить график* в формате **.grs*, это позволит в дальнейшем правильно оформить отчет с наложением графиков эксперимента. Не забывайте подписывать график, а также характеристики распределения исследуемой величины (математическое ожидание, дисперсию), поскольку они не сохраняются.

Таким образом, анализируя чувствительность и качество функционирования устойчивой САР, в данном параграфе мы изменяем вероятностные настройки регулятора (математическое ожидание, дисперсию) и получим вероятностные характеристики переходного процесса, как показатели качества устойчивой САР (время регулирования $t_р$, максимальное динамическое отклонение σ, перерегулирование η).

Для системы с ПИД-регулятором при проведении однофакторного эксперимента:

1) сначала задайте оптимальные значения математического ожидания параметров регулятора (базовая точка эксперимента);

2) затем изменяйте математическое ожидание коэффициента передачи регулятора $\mu_{k_p} = k_{\text{р}}^{\text{о}} \pm \Delta k_{\text{р}}$ при постоянных значениях математических ожиданий $\mu_{T_И} = T_{\text{И}}^{\text{о}}$ и $\mu_{T_Д} = T_{\text{Д}}^{\text{о}}$;

3) далее изменяйте математическое ожидание постоянной интегрирования $\mu_{T_И} = T_{\text{И}}^{\text{о}} \pm \Delta T_{\text{И}}$ при постоянных значениях математических ожиданий $\mu_{k_p} = k_{\text{р}}^{\text{о}}$ и $\mu_{T_Д} = T_{\text{Д}}^{\text{о}}$;

4) и наконец изменяйте математическое ожидание постоянной дифференцирования $\mu_{T_Д} = T_{\text{Д}}^{\text{о}} \pm \Delta T_{\text{Д}}$ при постоянных значениях математических ожиданий $\mu_{k_p} = k_{\text{р}}^{\text{о}}$ и $\mu_{T_И} = T_{\text{И}}^{\text{о}}$.

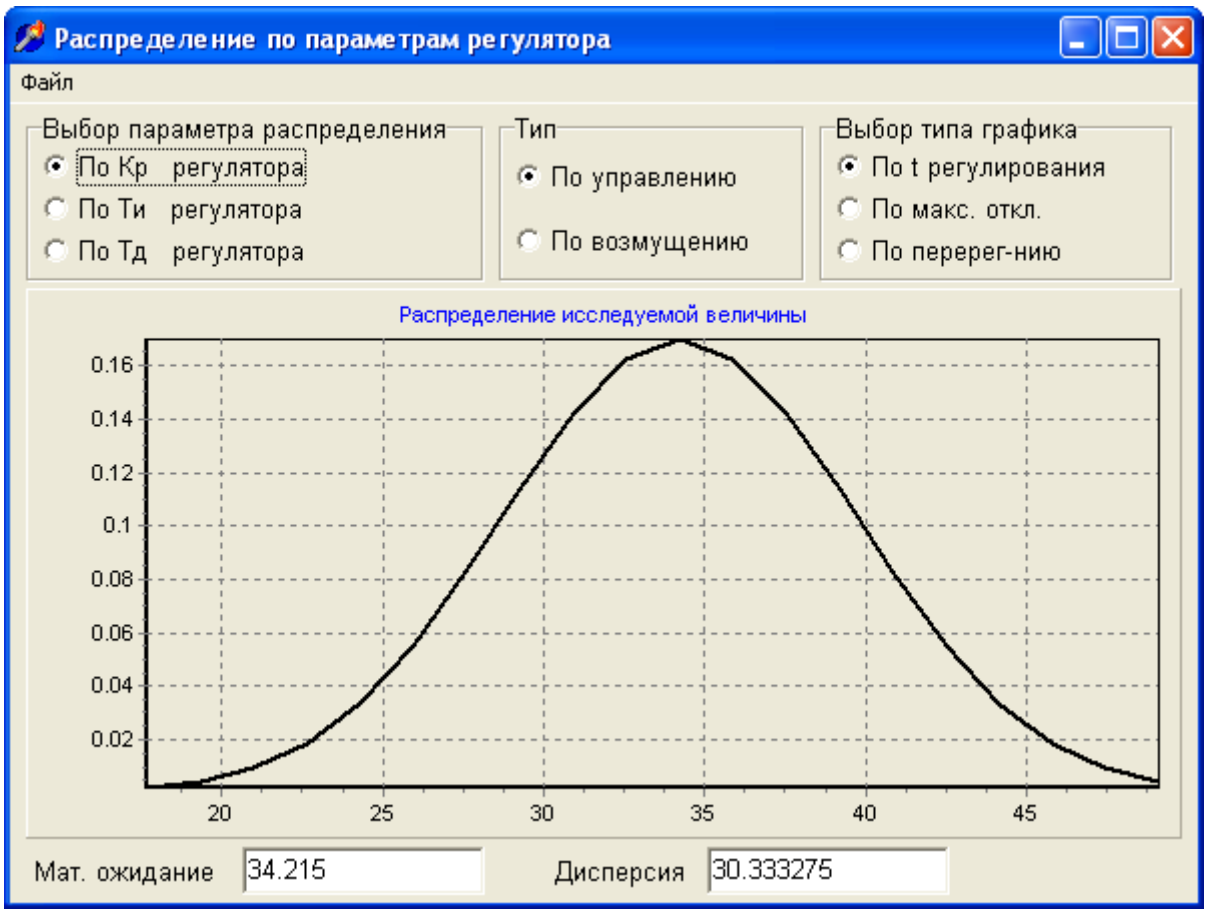


Рис. 2.77. Построение распределения исследуемой величины

При этом дисперсия настроек регулятора остается одинаковой и неизменной для всех параметров.

Каждый раз анализируйте результаты эксперимента и выявите закономерности влияния варьируемых параметров регулятора на показатели качества системы.

После проведения однофакторного эксперимента, например п. 1 и п. 2, т.е. при изменении математического ожидания коэффициента усиления

μ_{k_p} (при этом постоянная интегрирования $T_Д$ и постоянная дифференцирования $T_Д$ остаются постоянными) у вас получатся семейства характеристик:

– по 3 графика распределения времени регулирования $t_р$ для устойчивой САР по управлению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$;

– по 3 графика распределения максимального динамического отклонения σ для устойчивой САР по управлению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$;

– по 3 графика распределения перерегулирования η для устойчивой САР по управлению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$;

– по 3 графика распределения времени регулирования $t_р$ для устойчивой САР по возмущению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$;

– по 3 графика распределения максимального динамического отклонения σ для устойчивой САР по возмущению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$;

– по 3 графика распределения перерегулирование η для устойчивой САР возмущению при математических ожиданиях равных $k_р^o$, $k_р^o + \Delta k_р$, $k_р^o - \Delta k_р$.

Аналогично получатся семейства графиков при проведении п. 3 и п. 4 однофакторного эксперимента.

Для представления результатов однофакторного эксперимента в виде семейства характеристик воспользуйтесь меню *Отчет / Наложение графиков* (см. рис. 2.78). Сначала выберите тип графика *Разброс* (в формате **.grs*), затем, нажимая кнопку *Добавить*, вставьте соответствующий график. Если тип графика не соответствует, т.е. вместо добавления графика *Разброс* вы выбрали например *АФЧХ*, то график не будет вставлен. Обязательно сразу сохраните каждое семейство графиков в формате, например, **.bmp*.

Для эффективности написания отчета по самостоятельной работе сразу подписывайте графики, вписывайте значения математического ожидания и дисперсии полученной характеристики, а также указывайте названия и значения осей.

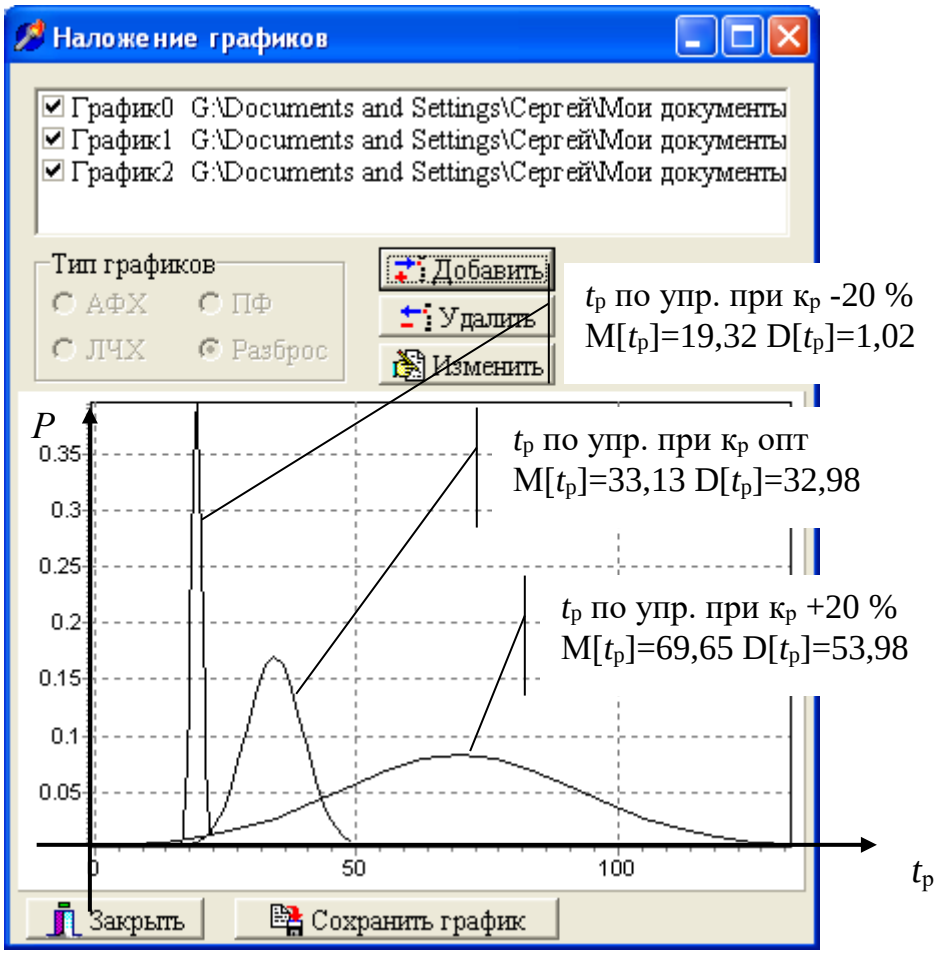


Рис. 2.78. Семейства характеристик распределения времени регулирования $t_р$ для устойчивой САР по управлению при математических ожиданиях равных $k_р^о$, $k_р^о + \Delta k_р$, $k_р^о - \Delta k_р$

2.6.4. Задания для самостоятельной работы

1. Запустить программу *C:\USERS\MS\Sianrg.exe*, которая используется для моделирования САР.

2. Составить параметрическую модель САР, для чего следует найти базовые (оптимальные) настройки для ПИД-регулятора $k_р^о$, $T_И^о$ и $T_Д^о$ методом Циглера-Никольса. Данные по передаточной функции (2.8) объекта управления $k_{об}$, $T_{об}$ и $\tau_{об}$ взять из табл. 2.8. Номер варианта уточнить у преподавателя.

3. Провести анализ показателей качества САР при разбросе параметров (настроек регулятора) системы. Зафиксировать графики распределений, математическое ожидание и дисперсию показателей качества для устойчивой САР по управлению и возмущению: время регулирования $t_р$, максимальное

динамическое отклонение σ, перерегулирование η. Далее эти параметры называются базовыми показателями качества СУ.

4. С целью определения чувствительности системы к стохастическим вариациям параметров регулятора провести однофакторный эксперимент. В качестве факторов эксперимента выбрать настройки регулятора $k_{\text{р}}$, $T_{\text{И}}$ и $T_{\text{Д}}$. Провести 2 эксперимента.

4.1. При статистическом задании параметров необходимо изменять вероятностные настройки ПИД-регулятора, а именно математическое ожидание, дисперсию оставить по умолчанию равной 0,1.

Для этого необходимо сначала задать коэффициент передачи регулятора с математическим ожиданием $\mu_{k_p} = k_{\text{р}}^{\text{о}} \pm \Delta k_{\text{р}}$ при постоянных значениях параметров с математическим ожиданием $\mu_{T_И} = T_{\text{И}}^{\text{о}}$ и $\mu_{T_Д} = T_{\text{Д}}^{\text{о}}$. Затем задать постоянную интегрирования с математическим ожиданием $\mu_{T_И} = T_{\text{И}}^{\text{о}} \pm \Delta T_{\text{И}}$ при постоянных значениях $\mu_{k_p} = k_{\text{р}}^{\text{о}}$ и $\mu_{T_Д} = T_{\text{Д}}^{\text{о}}$. Наконец изменять постоянную дифференцирования с математическим ожиданием $\mu_{T_Д} = T_{\text{Д}}^{\text{о}} \pm \Delta T_{\text{Д}}$ при постоянных значениях параметров с математическим ожиданием $\mu_{k_p} = k_{\text{р}}^{\text{о}}$ и $\mu_{T_И} = T_{\text{И}}^{\text{о}}$.

Уровень вариации параметров регулятора Δ может быть равен ±10, ±20, ±30 % (уточнить у преподавателя).

Результаты однофакторного эксперимента для устойчивой САР по управлению и возмущению представить в виде семейства характеристик показателей качества время регулирования $t_{\text{р}}$, максимальное динамическое отклонение σ, перерегулирование η. Семейства характеристик изобразить по 3 графика распределения при математических ожиданиях равных

$$k_{\text{р}}^{\text{о}},\ k_{\text{р}}^{\text{о}} + \Delta k_{\text{р}},\ k_{\text{р}}^{\text{о}} - \Delta k_{\text{р}};$$

$$T_{\text{И}}^{\text{о}},\ T_{\text{И}}^{\text{о}} + \Delta T_{\text{И}},\ T_{\text{И}}^{\text{о}} - \Delta T_{\text{И}};$$

$$T_{\text{Д}}^{\text{о}},\ T_{\text{Д}}^{\text{о}} + \Delta T_{\text{Д}},\ T_{\text{Д}}^{\text{о}} - \Delta T_{\text{Д}}).$$

Обязательно зафиксировать математическое ожидание и дисперсию показателей качества.

4.2. При статистическом задании параметров необходимо изменять вероятностные настройки ПИД-регулятора, а именно дисперсию, математическое ожидание оставить по умолчанию равной оптимальной настройке.

Для этого необходимо сначала задать дисперсию коэффициента передачи регулятора $\sigma_{k_p}^2 = 0{,}1 \pm \Delta$ при постоянных значениях дисперсий (заданных по

умолчанию) других параметров $\sigma^2_{T_И} = 0{,}1$ и $\sigma^2_{T_Д} = 0{,}1$. Затем задать дисперсию постоянной интегрирования $\sigma^2_{T_И} = 0{,}1 \pm \Delta$ при постоянных значениях дисперсий (заданных по умолчанию) других параметров $\sigma^2_{k_p} = 0{,}1$ и $\sigma^2_{T_Д} = 0{,}1$. Наконец изменять дисперсию постоянной дифференцирования $\sigma^2_{T_Д} = 0{,}1 \pm \Delta$ при постоянных значениях дисперсий (заданных по умолчанию) других параметров $\sigma^2_{k_p} = 0{,}1$ и $\sigma^2_{T_И} = 0{,}1$.

Уровень вариации параметров регулятора Δ может быть равен ±20, ±50, ±90 % (уточнить у преподавателя).

Результаты однофакторного эксперимента для устойчивой САР по управлению и возмущению представить в виде семейства характеристик показателей качества: время регулирования $t_р$, максимальное динамическое отклонение σ, перерегулирование η. Семейства характеристик изобразить по 3 графика распределения при дисперсиях параметров регулятора равных

$$\sigma^2_{k_p} = 0{,}1,\ \sigma^2_{k_p} = 0{,}1 + \Delta,\ \sigma^2_{k_p} = 0{,}1 - \Delta;$$

$$\sigma^2_{T_И} = 0{,}1,\ \sigma^2_{T_И} = 0{,}1 + \Delta,\ \sigma^2_{T_И} = 0{,}1 - \Delta;$$

$$\sigma^2_{T_Д} = 0{,}1,\ \sigma^2_{T_Д} = 0{,}1 + \Delta,\ \sigma^2_{T_Д} = 0{,}1 - \Delta.$$

Обязательно зафиксировать математическое ожидание и дисперсию показателей качества.

6. Результаты синтеза и анализа САР представить следующим образом:

6.1. Исходные данные ОУ, базовые (оптимальные) настройки для ПИД-регулятора $k_р^о$, $T_И^о$ и $T_Д^о$, а также базовые показатели качества САР, полученные при разбросе параметров (настроек регулятора) системы: графики распределений, математическое ожидание и дисперсию показателей качества для устойчивой САР по управлению и возмущению: время регулирования $t_р$, максимальное динамическое отклонение σ, перерегулирование η.

6.2. Исходные данные и результаты однофакторных экспериментов в виде семейства характеристик показателей качества (см. задание п.4.2. и п. 4.3) для устойчивой САР по управлению и возмущению: время регулирования $t_р$, максимальное динамическое отклонение σ, перерегулирование η, с указанием на графиках математических ожиданий и дисперсий показателей качества.

6.3. Выявить и написать выводы о закономерности влияния варьируемых параметров регулятора на показатели качества системы.

2.7. КОРРЕЛЯЦИОННЫЙ, РЕГРЕССИОННЫЙ И ДИСПЕРСИОННЫЙ АНАЛИЗ СИСТЕМ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ

2.7.1. Особенности обработки и анализа результатом машинного моделирования

2.7.1.1. Оценка характеристик стохастической модели

Успех имитационного эксперимента с моделью системы существенным образом зависит от правильного решения вопросов обработки и последующего анализа и интерпретации результатов моделирования.

В качестве оценок для искомых характеристик при проведении имитационного эксперимента со стохастической моделью рассчитывают средние значения математического ожидания, дисперсии, корреляционные моменты и т. д. (см. п. 2.6.2). При обработке результатов машинного эксперимента с моделью наиболее часто возникают следующие задачи: определение эмпирического закона распределения случайной величины, проверка однородности распределений, сравнение средних значений и дисперсий переменных, полученных в результате моделирования, и т.д. Эти задачи с точки зрения математической статистики являются типовыми задачами по проверке статистических гипотез.

Задача определения эмпирического закона распределения случайной величины – наиболее общая из перечисленных, но для правильного решения требует большого числа реализации N. В этом случае по результатам машинного эксперимента находят значения выборочного закона распределения $F_э(y)$ (или функции плотности $f_э(y)$) и выдвигают нулевую гипотезу H_0, что полученное эмпирическое распределение согласуется с каким-либо теоретическим распределением. Проверяют эту гипотезу H_0 с помощью статистических критериев согласия Колмогорова, Пирсона, Смирнова, Фишера и т.д., причем необходимую в этом случае статистическую обработку результатов ведут по возможности в процессе моделирования системы на ЭВМ.

Для принятия или опровержения гипотезы выбирают некоторую случайную величину U, характеризующую степень расхождения теоретического и эмпирического распределения, связанную с недостаточностью статистического материала и другими случайными причинами. Закон распределения этой случайной величины зависит от закона распределения случайной величины η и числа реализации N при статистическом моделировании системы. Если вероятность расхождения теоретического и эмпирического распределения $P\{U_t \geq U\}$ велика в понятиях применяемого критерия согласия, проверяемая гипотеза о виде распределения H_0 не опровергается. Выбор вида теоретического распределения $F(y)$ (или $f(y)$) проводится по графикам (гистограммам) $F_э(y)$ (или $f_э(y)$), выведенным на печать или на экран дисплея.

Хотя рассмотренные оценки искомых характеристик процесса функционирования системы, полученные в результате машинного эксперимента с моделью, являются простейшими, но охватывают большинство случаев, встречающихся в практике обработки результатов моделирования системы для целей ее исследования и проектирования.

2.7.1.2. Виды анализа результатов машинного моделирования

Существуют различные методы анализа, зависящие от целей исследования и вида получаемых при моделировании характеристик. Рассмотрим особенности использования методов корреляционного, регрессионного и дисперсионного анализа для результатов моделирования систем.

С помощью *корреляционного анализа* можно установить, насколько тесна связь между двумя (или более) случайными величинами, наблюдаемыми и фиксируемыми при моделировании конкретной системы *S*. Корреляционный анализ результатов моделирования сводится к оценке разброса значений η относительно среднего значения $\bar{y}$, т.е. к оценке силы корреляционной связи. Существование этих связей и их тесноту можно для схемы корреляционного анализа $y = M[\eta/\xi = x]$ выразить при наличии линейной связи между исследуемыми величинами и нормальности их совместного распределения с помощью коэффициента корреляции

$$r_{\xi\eta} = \frac{M[\xi - M[\xi]]M[\eta - M[\eta]]}{\sqrt{D[\xi]D[\eta]}} = \frac{M[\xi - \mu_\xi]M[\eta - \mu_\eta]}{\sigma_\xi \sigma_\eta}, \qquad (2.46)$$

т.е. второй смешанный центральный момент делится на произведение средних квадратичных отклонений, чтобы иметь безразмерную величину, инвариантную относительно единиц измерения рассматриваемых случайных переменных.

Связь между величиной коэффициента корреляции и оценкой тесноты связи по шкале Чеддока: 0,10 – 0,29 – слабая характеристика силы связи, 0,30 – 0,49 – умеренная, 0,50 – 0,69 – заметная, 0,70 – 0,89 – высокая, 0,90 – 1,00 – весьма высокая.

Коэффициент корреляции при *N* реализациях:

$$\tilde{r}_{\xi\eta} = \frac{\sum_{k=1}^{N}(x_k - \bar{x})(y_k - \bar{y})}{\left[\sum_{k=1}^{N}(x_k - \bar{x})^2 \sum_{k=1}^{N}(y_k - \bar{y})^2\right]^{1/2}} = \frac{\sum_{k=1}^{N} x_k y_k - N\bar{x}\bar{y}}{\left[\left(\sum_{k=1}^{N} x_k^{\,2} - N\bar{x}^2\right)\left(\sum_{k=1}^{N} y_k^{\,2} - N\bar{y}^2\right)\right]^{1/2}}. \qquad (2.47)$$

Данное соотношение требует минимальных затрат машинной памяти на обработку результатов моделирования. Получаемый при этом коэффициент корреляции $|r_{\xi\eta}| \le 1$. При сделанных предположениях $r_{\xi\eta} = 0$ свидетельствует о взаимной независимости случайных переменных ξ и η, исследуемых при моделировании (рис. 2.79, *а*). При $r_{\xi\eta} = 1$ имеет место функциональная

(т.е. нестохастическая) линейная зависимость вида $y = b_0 + b_1 x$, причем если $r_{\xi\eta} > 0$, то говорят о положительной корреляции, т.е. большие значения одной случайной величины соответствуют большим значениям другой (рис. 2.79, *б*). Случай $0<r_{\xi\eta}<1$ соответствует наличию линейной корреляции с рассеянием (рис. 2.79, *в*) либо наличию нелинейной корреляции результатов моделирования (рис. 2.79, *г*).

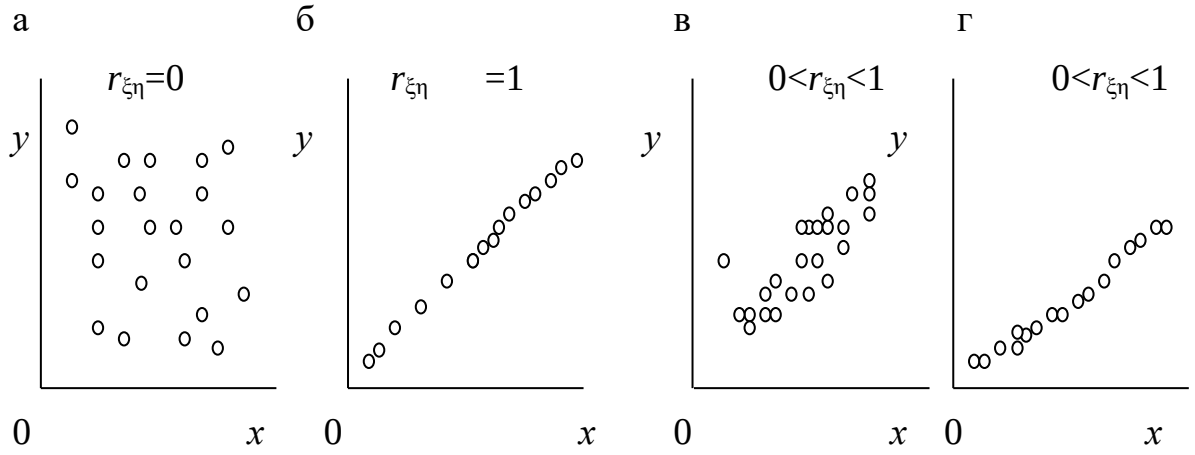


Рис. 2.79. Различные случаи корреляции переменных

Для того чтобы оценить точность полученной при обработке результатов моделирования системы *S* оценки $r_{\xi\eta}$, целесообразно ввести в рассмотрение коэффициент $w = \frac{1}{2}\ln\frac{1+r_{\xi\eta}}{1-r_{\xi\eta}}$, причем *w* приближенно подчиняется гауссовскому распределению со средним значением и дисперсией

$$\mu_w = \frac{1}{2}\ln\frac{1+r_{\xi\eta}}{1-r_{\xi\eta}}, \quad \sigma_w^2 = \frac{1}{N-3}. \tag{2.48}$$

При анализе результатов моделирования системы *S* важно отметить то обстоятельство, что если даже удалось установить тесную зависимость между двумя переменными, то отсюда еще не следует их причинно-следственная взаимообусловленность. Возможна ситуация, когда случайные ξ и η стохастически зависимы, хотя причинно они являются для системы *S* независимыми. При статистическом моделировании наличие такой зависимости может иметь место, например, из-за коррелированности последовательностей псевдослучайных чисел, используемых для имитации событий, положенных в основу вычисления значений *x* и *y*.

Таким образом, корреляционный анализ устанавливает связь между исследуемыми случайными переменными машинной модели и оценивает тесноту этой связи. Однако в дополнение к этому желательно располагать моделью зависимости, полученной после обработки результатов моделирования.

Регрессионный анализ дает возможность построить модель, наилучшим образом соответствующую набору данных, полученных в ходе машинного эксперимента с системой *S*. Под наилучшим соответствием понимается минимизированная функция ошибки, являющаяся разностью между прогнозируемой моделью и данными эксперимента. Такой функцией ошибки при регрессионном анализе служит сумма квадратов ошибок.

Рассмотрим особенности регрессионного анализа результатов моделирования при построении линейной регрессионной модели. На рис. 2.80, *а* показаны точки x_i, y_i, $i=\overline{1,N}$, полученные в машинном эксперименте с моделью системы *S*. Делаем предположение, что модель результатов машинного эксперимента графически может быть представлена в виде прямой линии

$$\hat{y} = \varphi(x) = b_0 + b_1 x, \qquad (2.49)$$

где $\hat{y}$ – величина, предсказываемая регрессионной моделью.

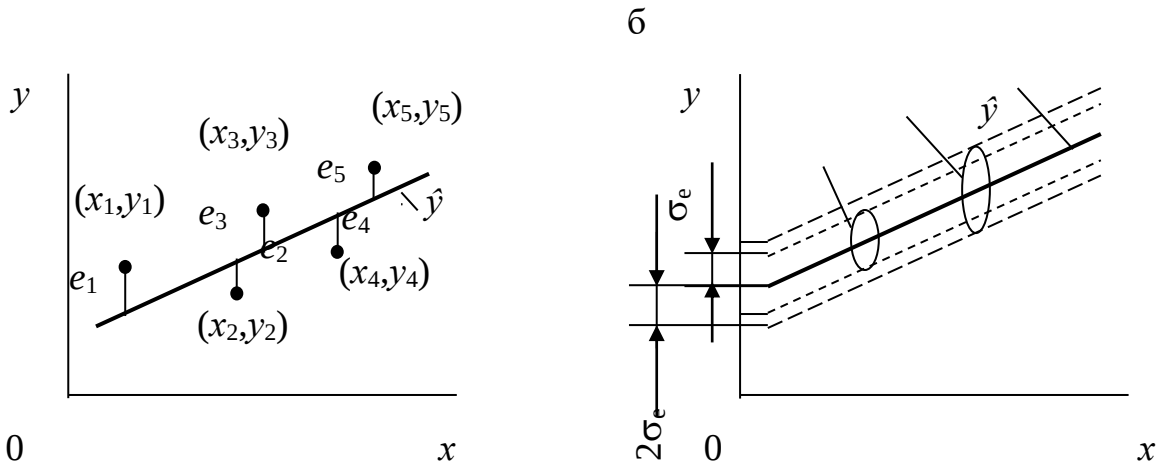


Рис. 2.80. Построение линейной регрессионной модели

Требуется получить такие значения коэффициентов b_0 и b_1, при которых сумма квадратов ошибок модели является минимальной. На рисунке ошибка e_i, $i=\overline{1,N}$ для каждой экспериментальной точки определяется как расстояние по вертикали от этой точки до линии регрессии $\hat{y} = \varphi(x)$.

Обозначим $\hat{y}_i = b_0 + b_1 x_i$, $i=\overline{1,N}$. Тогда выражение для ошибок будет иметь вид:

$e_i = \hat{y}_i - y_i = b_0 + b_1 x_i - y_i$, а функция ошибки $F_0 = \sum_{i=1}^{N} (b_0 + b_1 x_i - y_i)^2$.

Для получения b_0 и b_1, при которых функция F_0 является минимальной, применяются обычные методы математического анализа. Условием минимума является $\partial F_0/\partial b_0 = 0$, $\partial F_0/\partial b_1 = 0$.

Дифференцируя F_0, получаем

$$\frac{\partial F_0}{\partial b_0} = \frac{\partial \sum_{i=1}^{N} (b_0 + b_1 x_i - y_i)^2}{\partial b_0} = 2(Nb_0 + b_1 \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i) = 0,$$
$$\frac{\partial F_0}{\partial b_1} = \frac{\partial \sum_{i=1}^{N} (b_0 + b_1 x_i - y_i)^2}{\partial b_1} = 2(b_0 \sum_{i=1}^{N} x_i + b_1 \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} x_i y_i) = 0. \quad (2.50)$$

Решая систему этих двух линейных алгебраических уравнений, можно получить значения b_0 и b_1. В матричном представлении эти уравнения имеют вид:

$$\begin{Vmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{Vmatrix} \bullet \begin{Vmatrix} b_0 \\ b_1 \end{Vmatrix} = \begin{Vmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i y_i \end{Vmatrix}.$$

Решая это уравнение, получаем

$$b_0 = \frac{\sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i y_i}{N \sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2}, \quad (2.51)$$

$$b_1 = \frac{N \sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N \sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2}, \quad (2.52)$$

где N – число реализаций при моделировании системы.

Соотношения для вычисления b_0 и b_1 требуют минимального объема памяти ЭВМ для обработки результатов моделирования. Обычно мерой ошибки регрессионной модели служит среднее квадратичное отклонение

$$\sigma_e = \left[\frac{1}{N-2} \sum_{i=1}^{N} e_i^2 \right]^{1/2} = \left[\frac{1}{N-2} \sum_{i=1}^{N} (b_0 + b_1 x_i - y_i)^2 \right]^{1/2}. \quad (2.53)$$

Для нормально распределенных процессов приблизительно 67 % точек находится в пределах одного отклонения σ_e от линии регрессии и 95 % – в пределах $2\sigma_e$ (трубки A и B соответственно на рис. 2.80, *б*). Для проверки точности оценок b_0 и b_1 в регрессионной модели могут быть использованы, например, критерии Фишера (*F*-распределение) и Стьюдента (*t*-распределение). Аналогично могут быть оценены коэффициенты уравнения регрессии и для случая нелинейной аппроксимации.

При обработке и анализе результатов моделирования часто возникает задача сравнения средних выборок. Если в результате такой проверки окажется,

что математическое ожидание совокупностей случайных переменных $\{y^{(1)}\}$, $\{y^{(2)}\}$, …, $\{y^{(n)}\}$ отличается незначительно, то статистический материал, полученный в результате моделирования, можно считать однородным (в случае равенства двух первых моментов). Это дает возможность объединить все совокупности в одну и позволяет существенно увеличить информацию о свойствах исследуемой модели, а следовательно, и системы. Попарное использование для этих целей критериев Смирнова и Стьюдента для проверки нулевой гипотезы затруднено в связи с наличием большого числа выборок при моделировании системы. Поэтому для этой цели используется *дисперсионный анализ.*

Рассмотрим решение задачи дисперсионного анализа при обработке результатов моделирования системы в следующей постановке. Пусть генеральные совокупности случайной величины $\{y^{(1)}\}$, $\{y^{(2)}\}$, ..., $\{y^{(n)}\}$ имеют нормальное распределение и одинаковую дисперсию. Необходимо по выборочным средним значениям при некотором уровне значимости γ проверять нулевую гипотезу H_0 о равенстве математических ожиданий. Выявим влияние на результаты моделирования только одного фактора, т.е. рассмотрим однофакторный дисперсионный анализ.

Допустим, изучаемый фактор x привел к выборке значений неслучайной величины Y следующего вида: y_1, y_2, ..., y_k, где k – количество уровней x. Влияние фактора будем оценивать неслучайной величиной D_x, называемой *факторной дисперсией:*

$$D_x = \sigma_x^2 = \sum_{i=1}^{k} \frac{(y_i - \bar{y})^2}{k}, \qquad (2.54)$$

где $\bar{y}$ – среднее арифметическое значение величины Y.

Если генеральная дисперсия $D[y]$ известна, то для оценки случайности разброса наблюдений необходимо сравнить $D[y]$ с выборочной дисперсией $S_в^2$, используя критерий Фишера (F-распределение). Если эмпирическое значение $F_э$ попадает в критическую область, то влияние фактора x считается значимым, а разброс значений x – неслучайным.

Алгоритм применения критерия Фишера следующий:

1) вычисляется выборочное отношение $F_э = \sigma_1^2 / \sigma_2^2$;
2) определяется число степеней свободы $k_1 = N_1 - 1$ и $k_2 = N_2 - 1$, где N_1 и N_2 – объемы выборок для оценки σ_1^2 и σ_2^2 соответственно;
3) при выбранном уровне значимости γ по таблицам F-распределения находятся значения границ критической области

$$F_1 = \frac{1}{F_{1-\gamma/2}(k_1, k_2)}, \quad F_2 = F_{1-\gamma/2}(k_1, k_2);$$

4) проверяется неравенство $F_1 \le F_э \le F_2$; если это неравенство выполняется, то с доверительной вероятностью $\beta = 1 - \gamma$ нулевая гипотеза H_0: $\sigma_1^2 = \sigma_2^2$ (о принадлежности двух выборок к одной и той же генеральной совокупности) может быть принята.

Если генеральная дисперсия $D[y]$ до проведения машинного эксперимента с моделью неизвестна, то необходимо при моделировании найти ее оценку.

Пусть серия наблюдений на уровне y_i имеет вид: y_{i1}, y_{i2}, …, y_{in}, где n – число повторных наблюдений на i-м уровне. Тогда на i-м уровне среднее значение наблюдений

$$\bar{y}_i = \frac{1}{n}\sum_{j=1}^{n} y_{ij}\,, \tag{2.55}$$

а среднее значение наблюдений по всем уровням

$$\bar{y} = \frac{1}{kn}\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij} = \frac{1}{k}\sum_{i=1}^{k}\bar{y}_i\,. \tag{2.56}$$

Общая выборочная дисперсия всех наблюдений

$$S_в^2 = \frac{1}{kn-1}\left[\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij}^2 - \frac{1}{kn}\left(\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij}\right)^2\right]. \tag{2.57}$$

При этом разброс значений y определяется суммарным влиянием случайных причин и фактора x. Задача дисперсионного анализа состоит в том, чтобы разложить общую дисперсию $D[y]$ на составляющие, зависящие от случайных и неслучайных факторов.

Оценка генеральной дисперсии, связанной со случайными факторами, имеющая $k(n-1)$-ю степень свободы

$$\tilde{D}_0[y] = \frac{1}{k(n-1)}\left[\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij}^2 - \frac{1}{n}\sum_{i=1}^{k}\left(\sum_{j=1}^{n} y_{ij}\right)^2\right], \tag{2.58}$$

а оценка факторной дисперсии

$$\tilde{D}_x = \tilde{D}[y] - \tilde{D}_0[y]. \tag{2.59}$$

Учитывая, что факторная дисперсия наиболее заметна при анализе средних значений на i-м уровне фактора, а остаточная дисперсия (дисперсия случайности) для средних значений в n раз меньше, чем для отдельных измерений, найдем точную оценку выборочной дисперсии:

$$\tilde{D}_x + \frac{1}{n}\tilde{D}_0[y] = \frac{1}{k-1}\sum_{i=1}^{k}(y_i - \bar{y})^2\,. \tag{2.60}$$

Умножив обе части этого выражения на n, получим в правой части выборочную дисперсию $S_в^2$, имеющую $(k-1)$-ю степень свободы, численное

значение которой уже определено по (2.57). Тогда можно определить и оценку факторной дисперсии $\tilde{D}_x$, связанную с неслучайными факторами.

Согласно критерию Фишера, если при заданном уровне значимости γ выполняется неравенство $\frac{1}{F_{1-\gamma/2}(k_1,k_2)} < S_в^2 / \tilde{D}_0[y] < F_{1-\gamma/2}(k_1,k_2)$, то нулевую гипотезу H_0 о равенстве средних значений на различных уровнях можно считать справедливой и влиянием фактора x на результаты моделирования можно пренебречь. В противном случае влияние фактора x будет значимым.

Таким образом, дисперсионный анализ позволяет вместо проверки нулевой гипотезы о равенстве средних значений выборок проводить при обработке результатов моделирования проверку нулевой гипотезы о тождественности выборочной и генеральной дисперсии.

2.7.2. Анализ результатов моделирования систем автоматического регулирования с использованием программного модуля SIANRG

Программный модуль *SIANRG* предназначен для моделирования САР при варьировании параметров объекта управления (ОУ) и настроек регуляторов. Анализ САР позволяет определить устойчивость системы по амплитудно-фазовой характеристике разомкнутой САР, и качество системы по переходным характеристикам переходного процесса замкнутой САР. При определении чувствительности САР к параметрам ее элементов синтез и анализ повторяются многократно при изменении параметров детерминированно и стохастически (см. п. 2.5, 2.6).

Под действием на систему случайных факторов эксперимент проводится многократно, а обработка результатов машинного моделирования проводится с использованием регрессионного, корреляционного и дисперсионного анализов (рис. 2.81).

Используя результаты машинного моделирования САР, с помощью программного модуля *SIANRG* можно определить:

- влияние параметров объекта (постоянная времени ОУ $T_{об}$, коэффициент усиления ОУ $k_{об}$ и транспортное запаздывание ОУ $\tau_{об}$) на выходные параметры, показатели качества САР (время регулирования $t_р$, перерегулирование η и максимальное динамическое отклонение σ) и количественное соотношение между случайными величинами изучаемого процесса (регрессионный анализ);
- тесноту связи между качественными показателями САР (время регулирования $t_р$, перерегулирование η и максимальное динамическое отклонение σ), сравниваемыми попарно, и оценку разброса значений этих параметров относительно среднего значения, т.е. оценку силы корреляционной связи (корреляционный анализ);

- общую выборочную дисперсию S_b^2 и оценку генеральной дисперсии $D_0[y]$, связанную со случайными факторами, с помощью которых можно сделать вывод о случайной и неслучайной природе результатов моделирования САР (дисперсионный анализ).

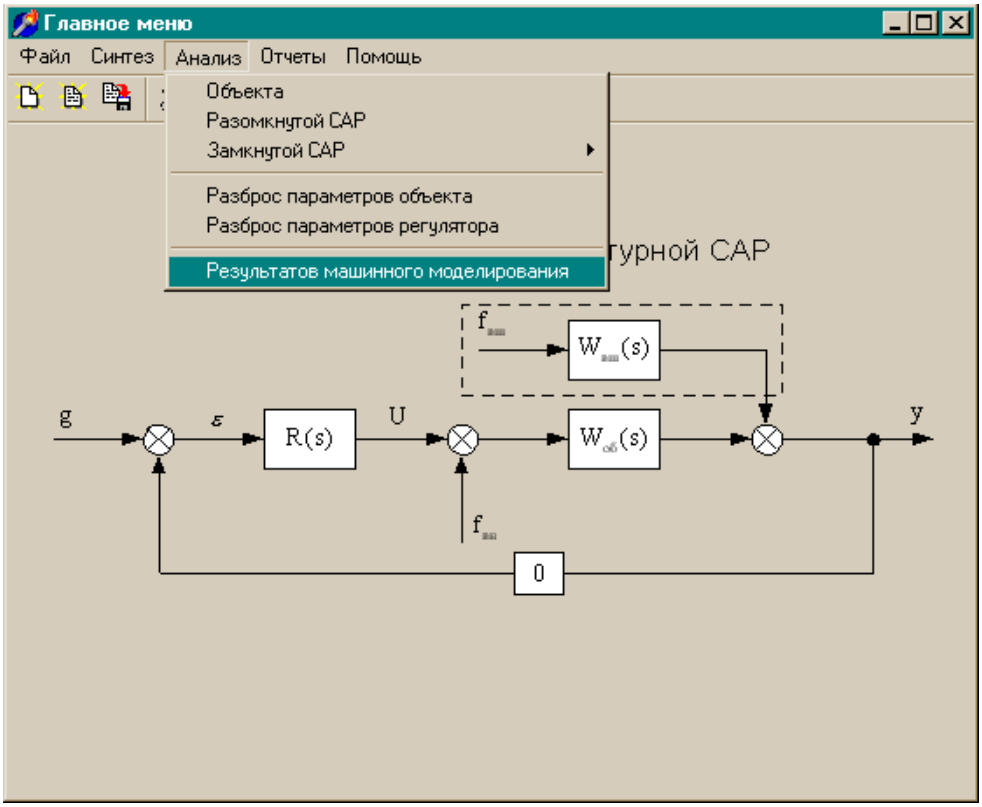


Рис. 2.81. Меню *Анализ* для обработки результатов машинного моделирования

Выбрать соответствующий вид анализа можно, используя кнопки с соответствующими надписями (рис. 2.82) или пункт меню *Тип* (см. рис. 2.83).

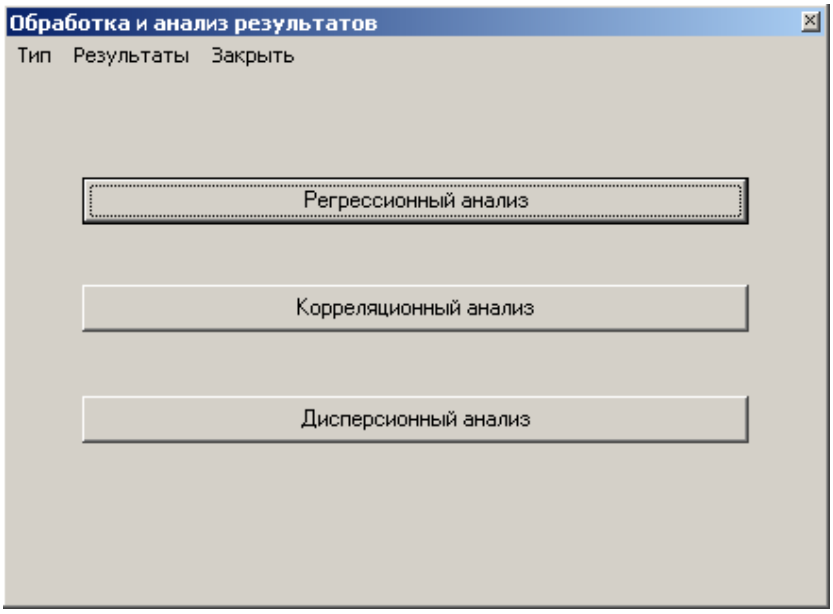


Рис. 2.82. Выбор вида анализа результатов машинного моделирования

Далее необходимо задать данные для обработки результатов.

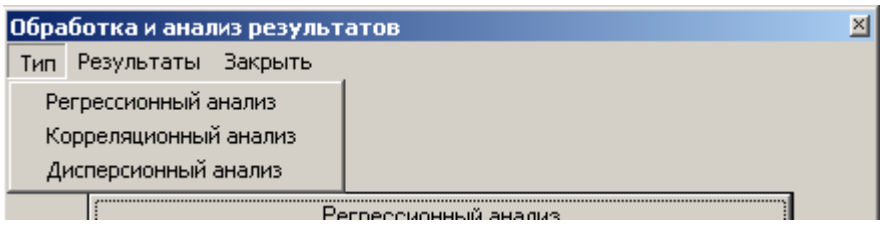


Рис. 2.83. Меню выбора типа анализа

Для задания данных регрессионного анализа (рис. 2.84) требуется ввести параметры ОУ $k_{об}$, $T_{об}$ и $\tau_{об}$, разбросы значений и числа итераций.

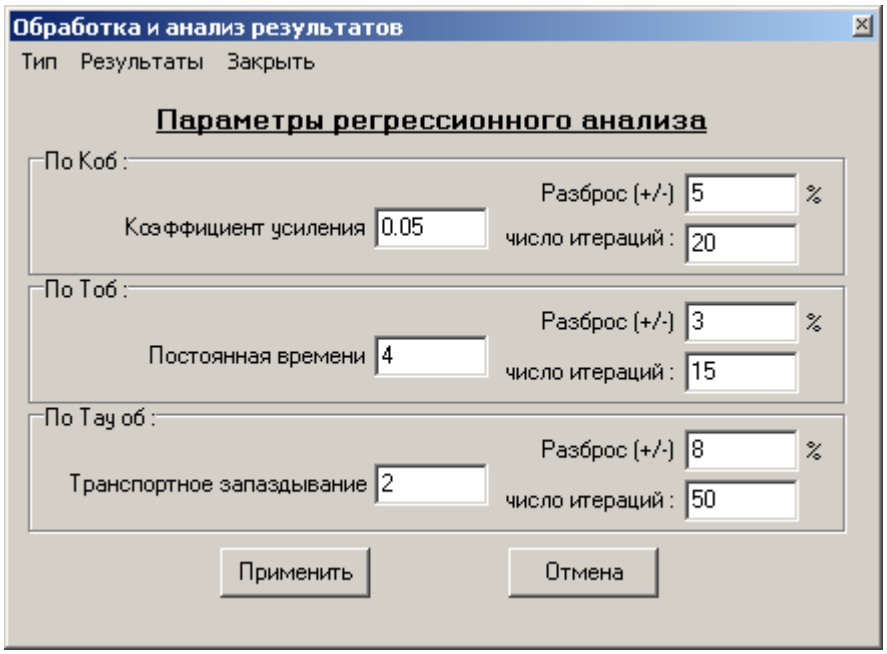


Рис. 2.84. Задание данных регрессионного анализа

Для задания данных для корреляционного анализа (см. рис. 2.85) вводятся значения математического ожидания и дисперсии параметров ОУ $k_{об}$, $T_{об}$ и $\tau_{об}$, величины желаемых разбросов и требуемых чисел итераций моделирования.

Ввод начальных данных дисперсионного анализа (см. рис. 2.86) предполагает задание математического ожидания и дисперсии параметров ОУ $k_{об}$, $T_{об}$ и $\tau_{об}$, числа повторных наблюдений n на i–м уровне и количества опытов k.

При загрузке форм передаются последние заданные при синтезе величины параметров ОУ (математические ожидания $k_{об}$, $T_{об}$ и $\tau_{об}$), остальные величины задаются по умолчанию. Для сохранения данных необходимо нажать кнопку *Применить*.

Для получения результатов машинного моделирования необходимо в пункте меню *Результаты* выбрать соответствующий вид анализа (см. рис. 2.87), что приведет к выводу соответствующих форм (см. рис. 2.88 – 2.90).

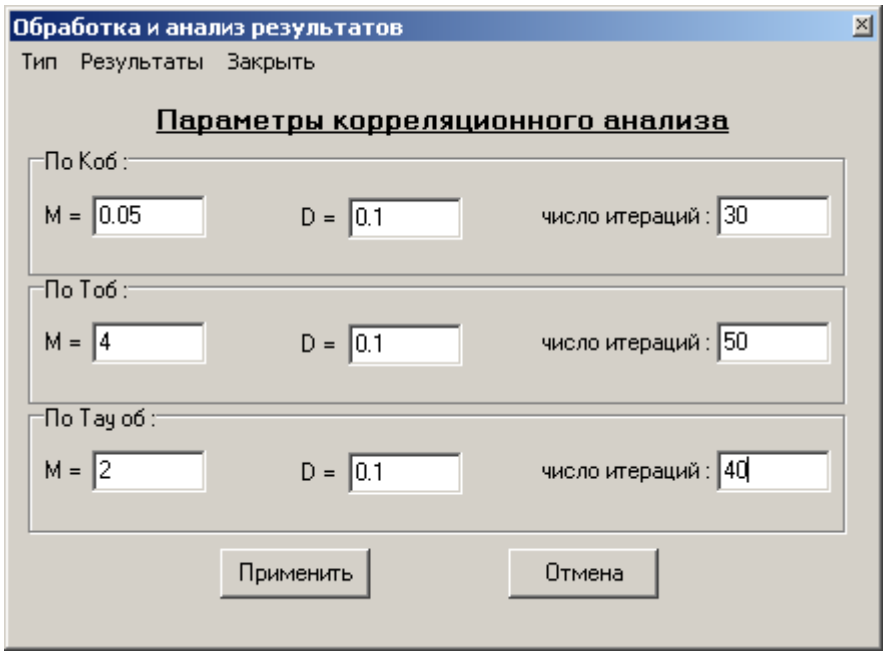


Рис. 2.85. Задание данных корреляционного анализа

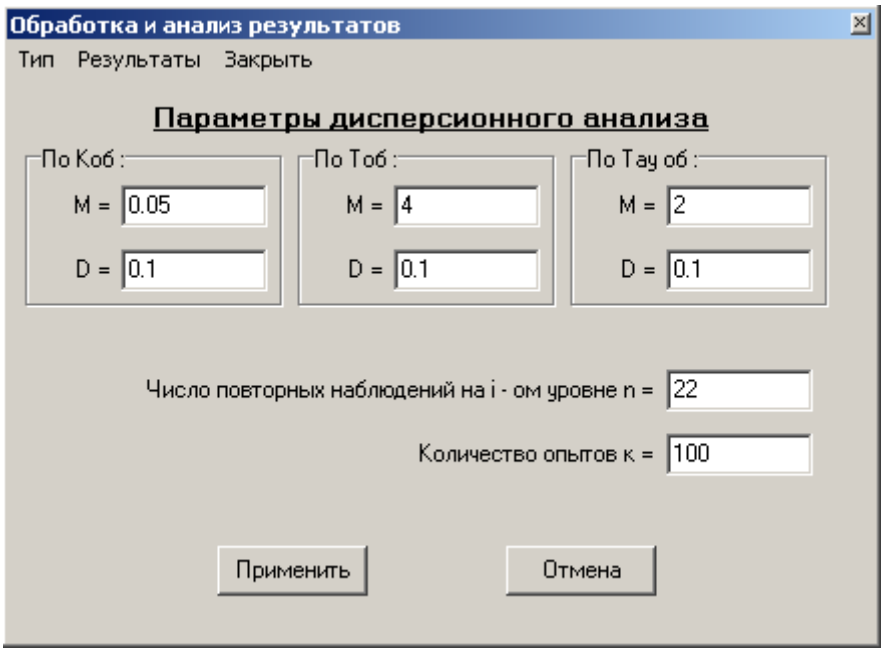


Рис. 2.86. Задание данных дисперсионного анализа

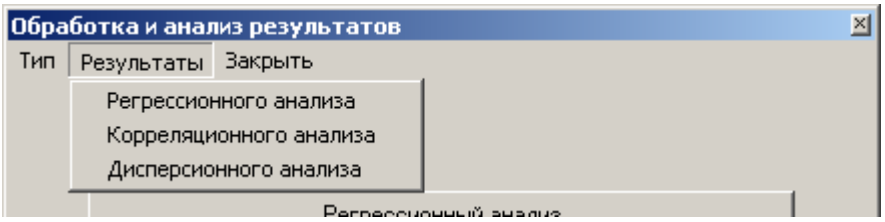


Рис. 2.87. Меню выбора результатов соответсвующего анализа

Выбор необходимых зависимостей характеристик процесса от параметров ОУ осуществляется с помощью соответствующих кнопок. Графические представления результатов работы могут быть сохранены при помощи кнопки *Сохранить график* с целью упрощения написания отчетов.

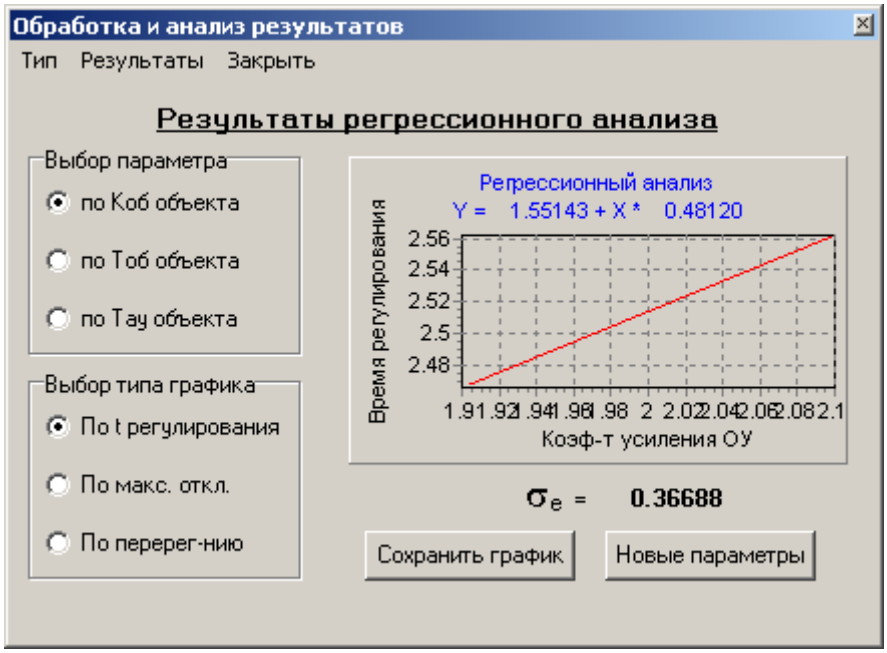


Рис. 2.88. Вывод результатов регрессионного анализа

Рис. 2.89. Вывод результатов корреляционного анализа

Результаты обработки представлены в виде графиков (для корреляционного и регрессионного анализов) и числовых значений (для корреляционного и дисперсионного анализов).

По результатам дисперсионного анализа пользователь должен принять или опровергнуть нулевую гипотезу H_0 о равенстве средних значений на различных

уровнях. Границы критической области определяют по таблицам *F*-распределения при выбранном уровне значимости γ = 5 – 10 %.

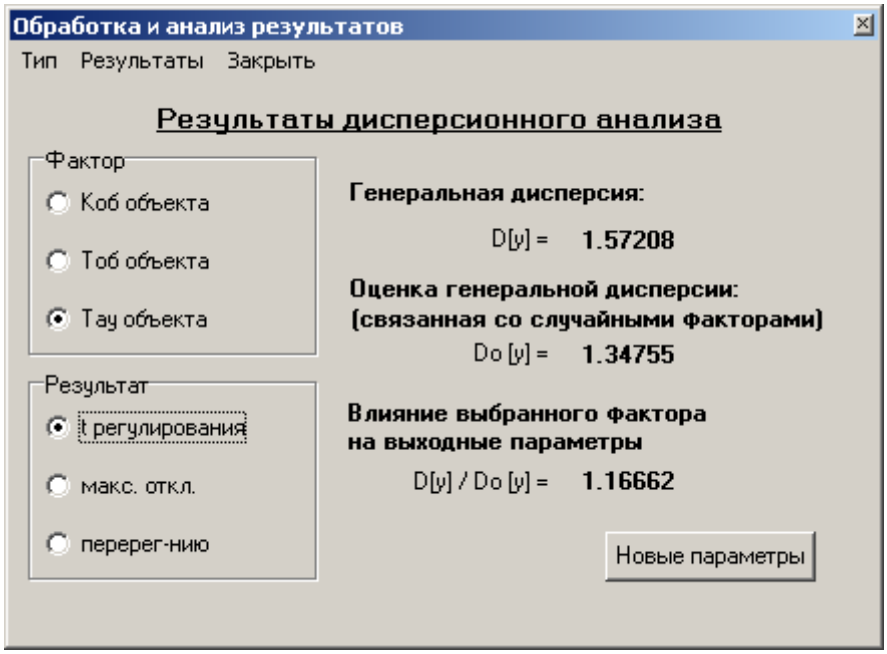


Рис. 2.90. Вывод результатов дисперсионного анализа

Если введенные данные нулевые или не соответствуют формату данных, или процесс в САР – расходящийся, то пользователь получит соответствующие сообщения (рис. 2.91, 2.92).

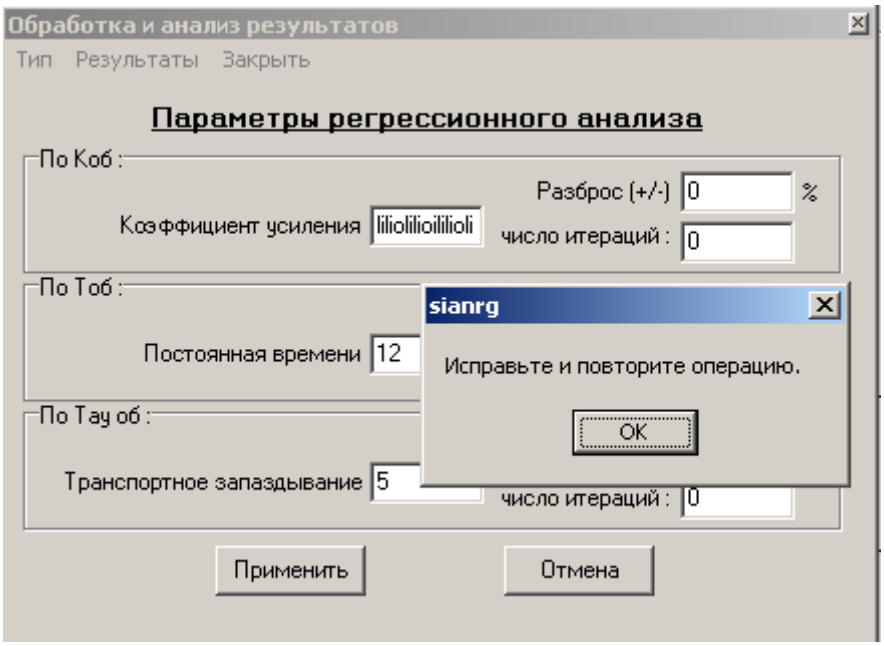


Рис. 2.91. Вывод сообщения о некорректно введенных данных

После завершения расчетов пользователь, не закрывая программного модуля, может изменить исходные данные для новых анализов либо снова приступить к синтезу и анализу модели.

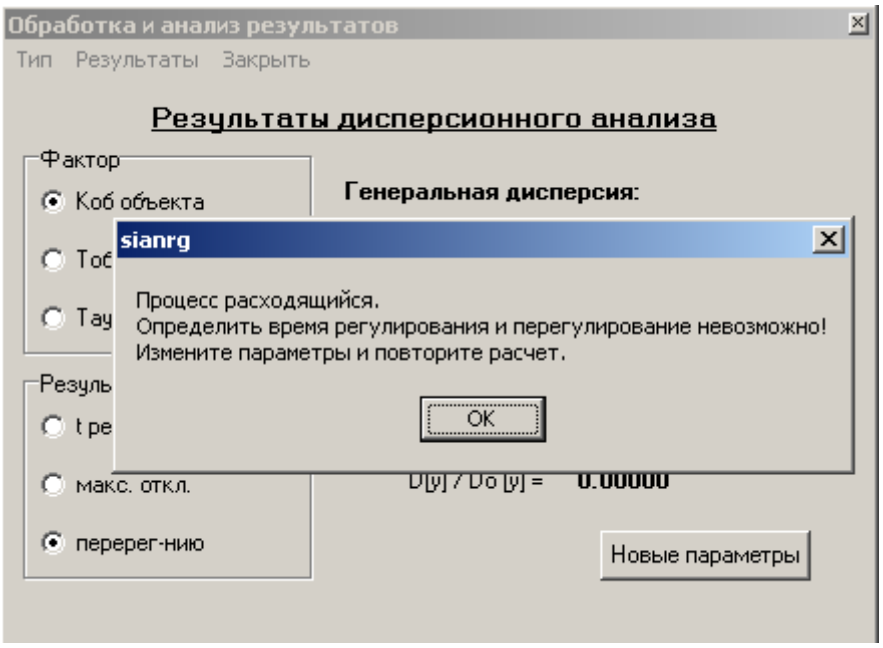


Рис. 2.92. Вывод сообщения о расходимости процесса

2.7.3. Задания для самостоятельной работы

1. Запустить программу *C:\USERS\MSU\Sianrg.exe*, которая используется для моделирования САР.

2. Составить параметрическую модель САР, для чего следует найти базовые (оптимальные) настройки П-, ПИ- и ПИД-регуляторов $k_{\text{р}}^{\text{о}}$, $T_{\text{И}}^{\text{о}}$ и $T_{\text{Д}}^{\text{о}}$. Данные по передаточной функции (2.8) ОУ $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ можно выбрать из табл. 2.8. Номер варианта уточнить у преподавателя. Определить характеристики качества замкнутой САР во временной области: время регулирования $t_{\text{р}}$, максимальное динамическое отклонение σ и перерегулирование η.

3. Провести обработку и анализ результатов стохастического моделирования замкнутой САР:

- для регрессионного анализа определить зависимости времени регулирования $t_{\text{р}}$, максимального динамического отклонения σ и перерегулирования η от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ отдельно, изменяя разброс (±5%, ±10%) и число итераций (20, 50, 100);

- для корреляционного анализа определить попарные зависимости времени регулирования $t_{\text{р}}$ от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$, максимального динамического отклонения σ от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$, перерегулирования η от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$, изменяя дисперсии $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ и число итераций (20, 50, 100);

- для дисперсионного анализа определить генеральные дисперсии, оценку генеральных дисперсий и влияние выбранного фактора ($k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$) на выходные параметры ($t_{\text{р}}$, σ, η), изменяя дисперсии $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$; число повторных наблюдений на i-м уровне n = 10, 20 и количество опытов k = 2, 10.

4. Результаты моделирования САР представить следующим образом:

- исходные данные ОУ, полученные базовые настройки П-, ПИ- и ПИД-регуляторов ($k_{\text{р}}^{\text{о}}$, $T_{\text{И}}^{\text{о}}$ и $T_{\text{Д}}^{\text{о}}$), а также базовые выходные параметры переходного процесса (время регулирования, максимальное динамическое отклонение, перерегулирование);

- семейство графиков по результатам обработки регрессионного анализа: зависимости времени регулирования $t_{\text{р}}$, максимального динамического отклонения δ и перерегулирования η от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ отдельно при изменении разброса ($\pm 5\%$, $\pm 10\%$) и числа итераций (20, 50, 100);

- семейство графиков по результатам обработки корреляционного анализа: попарные зависимости времени регулирования $t_{\text{р}}$ от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$, максимального динамического отклонения σ от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$, перерегулирования η от $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ при изменении дисперсии $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$ и числа итераций (20, 50, 100);

- результаты обработки дисперсионного анализа: генеральные дисперсии, оценку генеральных дисперсий и влияние выбранного фактора ($k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$) на выходные параметры ($t_{\text{р}}$, σ, η) при изменении дисперсии $k_{\text{об}}$, $T_{\text{об}}$ и $\tau_{\text{об}}$; число повторных наблюдений на i-м уровне $n = 10, 20$ и количество опытов $k = 2, 10$.

- выявить и написать выводы о закономерности влияния варьируемых параметров регулятора на показатели качества системы.

2.8. МОДЕЛИРОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ С ПОМОЩЬЮ ПАКЕТА VISSIM

2.8.1. Стандартные блоки пакета программ VISSIM

Пакет программ VISSIM (см. п. 2.3.) предназначен для моделирования процессов в системах управления объектами, которые можно описать дифференциальными уравнениями или передаточными функциями. Ниже более подробно описаны функции блоков данного пакета программ.

2.8.1.1. Блоки аннотаций (Annotation)

Пакет VISSIM содержит следующие блоки аннотаций:

- Comment – блок комментария;
- Date – блок текущей даты и времени;
- ScalarToVec – группировка в одну линию нескольких соединительных линий;
- Variable – блок переменной;
- WireLabel – однострочный комментарий;
- VecToScalar – расщепление группового соединения;

- WirePositioner – блок позиционирования соединений на экране для улучшения наглядности.

1. Annotation/comment – блок комментария.

Добавляет окно комментария к схеме. Для ввода текста в окно комментария щелкните ПК мыши на блоке. Курсор мыши изменится на I-знак (мигающий курсор в наборном поле блока), показывающий, что блок находится в режиме вставки. Для выхода из режима вставки снова щелкните ПК мыши на блоке.

2. Annotation/date – блок даты.

Отображает текущую дату и время в формате: день, номер месяца, ч, мин, с, год. Для корректировки времени или даты перейдите в System Control Panel. Этот блок не имеет параметров.

3. Annotation/scalarToVec – блок объединения сигналов.

Объединяет входные сигналы в одну векторную связь. Используйте Edit/Add Input и Edit/Remove Input для изменения числа входных меток. По умолчанию - три. Этот блок не имеет параметров.

4. Annotation/variable – блок переноса переменной.

Передает значения данных через схему без гибких связей. Блоки имеют названия совместно используемых сигналов. Только один блок, давший имя переменной, может иметь вход, но блоки переменной могут иметь несколько выходов. Переменная определяет имя блока. Если имя переменной начинается двоеточием (:), то это локальная переменная, на которую можно ссылаться только на текущем уровне схемы, что позволяет вам использовать блоки с одинаковым именем в различных составных блоках. Имеется три специальных имени переменных, встроенных в VISSIM:

- $firstPass – генерирует начальный единичный импульс на первом шаге моделирования;
- $lastPass – генерирует заключительный единичный импульс на последнем шаге моделирования;
- $runCount – содержит счетчик количества итераций при многократном выполнении моделирования. Используйте эту переменную с параметром автоматического запуска из Simulate/Change Parameters... для выполнения моделирования методом Monte Carlo и подстройки параметров.

5. Annotation/vecToScalar – блок разделения объединенных сигналов.

Разбивает векторную связь на скалярные выходные сигналы. Используйте команды Edit/Add Output и Edit/Remove Output для установления числа выходных меток. По умолчанию – три. Этот блок не имеет параметров.

6. Annotation/wireLabel – текстовый комментарий на поле модели.

Вставляет однострочный комментарий, соответствующий помеченной части схемы. Для установки или подавления обрамления используйте параметр Box WireLabels в диалоговом окне Edit/Preferences... WireLabel определяется комментарием.

7. Annotation/wirePositioner – блок для позиционирования проводов.

Позволяет располагать гибкие связи с минимальными нагромождениями на экране. Блок состоит из входной и выходной меток, соединенных гибкой связью. Они не требуют дополнительного времени на вычисления во время моделирования. Этот блок не имеет параметров.

2.8.1.2. Арифметические блоки (Arithmetic)

Пакет VISSIM содержит следующие арифметические блоки:

1/x – инверсия входной величины: выходной сигнал равен обратной величине входного: $y = 1/x$ (не путать с блоком деления).

–x – инвертор знака, знак выхода противоположен знаку входа: $y = -x$.

* – блок умножения, количество входов может быть любым: $y = x_1 \times x_2$.

/ – блок деления имеет два входа, его выход равен отношению величин верхнего и нижнего входных сигналов: $y = x_1/x_2$.

abs – блок, вычисляющий абсолютное значение входного сигнала: $y = | x |$.

gain – блок-усилитель с одним входом, выполняет умножение сигнала на константу, которая вводится в меню блока: $y = ax$.

pow – блок с двумя входами возведения первого сигнала x_1 в степень, равную величине второго сигнала x_2 (второй сигнал может отсутствовать, в этом случае степень по умолчанию равна двум): $y = x_1^{x_2}$.

sign – определяет знак входного сигнала: $y = \text{sign}(x)$.

SummingJunction – сумматор, количество входов и знаки слагаемых можно изменять: $y = x_1 + x_2 + ... + x_N + C$, имеет начальное смещение выхода C, которое устанавливается в меню блока.

2.8.1.3. Логические блоки (Boolean)

Пакет VISSIM содержит следующие логические блоки с двумя входами и одним выходом, который может принимать значения 0 или 1:

> (x_1 больше, чем x_2);

< (x_1 меньше, чем x_2);

≥ (x_1 больше или равно x_2);

≤ (x_1 меньше или равно x_2);

== (x_1 равно x_2);

!= (x_1 не равно x_2);

and – поразрядное «И» x_1 и x_2;

or – поразрядное «ИЛИ» x_1 и x_2;

xor – поразрядное исключительное «ИЛИ» x_1 и x_2.

Пакет VISSIM содержит логический блок с одним входом и одним выходом, который может принимать значения 0 или 1:

not – НЕ, логическое отрицание.

Булевы (логические) блоки не имеют параметров. Нажатием ПК мыши на любом булевом блоке можно выбрать нужное отношение.

2.8.1.4. Блоки интеграторов (Integration)

Пакет VISSIM содержит следующие блоки интегрирования:

- Integrator – простой интегратор;
- LimitedIntegrator – интегратор с пределами интегрирования;
- ResetIntegrator – интегратор со сбросом;
- TransferFunction – передаточная функция.

В этих блоках параметр Initial Condition показывает начальное значение выходной координаты интегратора, которое по умолчанию равно нулю. Параметр ID представляет идентификационный номер блока. Он содержит порядковый номер, который VISSIM назначает интегратору. По умолчанию – 0, максимум – 255 для Personal VISSIM и 16344 для VISSIM.

1. Integration/integrator – простой интегратор.

$$y = \int x .$$

Выполняет численное интегрирование входной величины. Используйте команду Simulate/Change Parameters... для выбора метода интегрирования.

2. Integration/limitedIntegrator – интегратор с пределами интегрирования.

$$y = \int_{x_3}^{x_2} x_1 .$$

Выполняет численное интегрирование входной величины, ограничивая внутренние состояния, а следовательно, и выход с возможностью установления верхней и нижней границ. Если значение интеграла достигает предела, то, как только производная изменит знак, интегрирование идет в обратную сторону от предела. Используйте команду Simulate/Change Parameters... для выбора метода интегрирования. Этот блок имеет три входа: x_1 – производная, x_2 – верхний предел, x_3 – нижний предел.

3. Integration/resetIntegrator (1/*S*) – интегратор со сбросом, выходной сигнал которого описывается следующим алгоритмом: если $y = | x_2 | <1$, то $y = \int x_1$; иначе $y = x_3$.

Блок выполняет численное интегрирование входной величины с возможностью сброса. Используйте команду Simulate/Change Parameters... для выбора метода интегрирования. Параметр Initial Condition может быть заблокирован, если на входе x_2 – не логический 0 на первом шаге моделирования.

4. Integration/transferFunction – передаточная функция.

Параметр 1: представляет передаточную функцию, определенную через полиномы числителя и знаменателя. Параметр Gain – указывает коэффициент передачи. VISSIM корректирует коэффициент передачи так, чтобы коэффициенты многочленов числителя и знаменателя при старших степенях равнялись единице. Окно Numerator служит для ввода коэффициентов многочлена числителя. Окно Denominator – для ввода коэффициентов многочлена знаменателя. Введите коэффициенты, отделяя их пробелами, начиная с самого высокого порядка. Не включайте знаки препинания или *s*-оператор. VISSIM определяет порядок передаточной функции по числу коэффициентов в знаменателе. Например, передаточная функция *n*-го порядка будет иметь *n*+1 коэффициентов знаменателя. Параметр Discrete – определяет дискретную передаточную функцию *Z*-области. По умолчанию – непрерывная *s*-область. Параметр *dT* определяет время для дискретной передаточной функции.

Параметр 2: представляет передаточную функцию, определенную как .MAT-файл. Когда блок читает .MAT-файл, то переводит матрицы пространства состояний A, B, C, D в SISO передаточную функцию *W*(*s*). При моделировании блок передаточной функции использует эти SISO передаточные функции *W*(*s*). VISSIM поддерживает только SISO определения. Use .MAT File показывает, что VISSIM будет выполнять передаточную функцию, определенную в .MAT-файле. Параметр MAT file определяет имя файла, который используется как вход для блока передаточной функции. Наберите имя файла в текстовом окне или щелкните ПК мыши на <Open. MAT File> для выбора из списка существующих файлов.

Ограничения: VISSIM поддерживает передаточные функции до 45-го порядка. Personal VISSIM поддерживает передаточные функции до 15-го порядка.

2.8.1.5. Блоки нелинейности (Nonlinear)

Пакет VISSIM содержит следующие блоки нелинейностей:

- CROSSDETECT – детектор пересечения сигналом заданного уровня;
- DEADBAND – зона нечувствительности;

- INT – округление (усечение до целого);
- LIMIT – блок ограничений заданными пределами;
- MAX – максимум входных сигналов;
- MERGE – условный выбор;
- MIN – минимум входных сигналов;
- QUANTIZE – квантователь входного сигнала;
- RELAY – оператор реле с двумя состояниями;
- SAMPLEHOLD – синхронизированная защелка входного сигнала;
- MAP – одно- или двухмерные кусочно-линейные отображения.

1. Nonlinear/crossDetect – детектор увеличения или уменьшения.

Выходная величина $y = 1$, если x пересекает величину, установленную в Cross Point, с положительным коэффициентом наклона; $y = –1$, если x пересекает величину, установленную в Cross Point, с отрицательным коэффициентом наклона; $y = 0$ в противном случае. По умолчанию величина в Cross Point равна 0.

2. Nonlinear/deadband – зона нечувствительности (мертвая зона).

Выдает выходной сигнал, равный входному сигналу, уменьшенному на зону «потери воздействия», где сигнал равен нулю. Этот блок используется для моделирования люфта в механических системах, таких как зубчатая передача или цепной механизм. Параметр Dead Band показывает ширину зоны нечувствительности. По умолчанию – 0,2. Выходной сигнал равен 0 в центре зоны.

3. Nonlinear/int – блок округления чисел с плавающей точкой до целых. Выходной сигнал равен целой части числа на входе (y = integer part x). Этот блок не имеет параметров.

4. Nonlinear/limit – блок насыщения (ограничитель уровня).

Если x меньше нижней границы, то y равен нижней границе, иначе если x больше верхней границы, то y равен верхней границе, иначе y равен x.

Блок ограничивает выходной сигнал в установленных верхнем и нижнем пределах. Параметр Lower Bound – наименьшая величина, которую может достигать выходной сигнал. По умолчанию – (–100). Параметр Upper Bound – наибольшая величина, которую может достигать выходной сигнал. По умолчанию – 100.

5. Nonlinear/map – кусочно-линейная нелинейность из .MAP-файла.

y = lookup (x) or y = lookup (x_1, x_2).

Задает одномерное или двухмерное кусочно-линейное отображение независимой переменной в одну или более зависимых переменных. Используйте блок вывода во внешний файл или любой редактор файла для

создания VisSim-файлов отображения. Зависимые переменные – линейная интерполяция значений независимых переменных между точками отображения и экстраполяция для значений вне пределов таблицы. Это свойство может использоваться для статической функциональной аппроксимации измеряемых данных или для калибровки устройства. Параметр Map File Name указывает имя многостолбцового файла, который сохраняет данные в текстовом формате ASCII. Нажмите кнопку <Open New File... >, чтобы выбрать нужный файл из списка существующих. Нажмите кнопку <Browse Data...> для чтения и редактирования выбранного файла отображения. Параметр Map Dimensions указывает размеры матрицы и правило формирования зависимой переменной (y).

При формировании зависимой переменной по правилу 1-D Mapping сопоставляет одну или более зависимых переменных одной независимой переменной. Файл данных должен располагаться так, чтобы в первом столбце чисел было множество независимых переменных в линейно увеличивающемся или уменьшающемся порядке, а каждый последующий столбец соответствовал зависимой переменной и каждая строка отображала независимую переменную на зависимую переменную. Используйте команду Edit/Add Output, чтобы добавить соединительные метки для каждой зависимой переменной в блоке отображения. Самая верхняя соединительная метка вывода соответствует крайнему левому столбцу зависимых переменных в таблице.

Правило формирования 2-D Mapping позволяет ставить в соответствие двум независимым переменным одну зависимую переменную. Соответствие ставится следующим образом: значения первой независимой переменой записываются в первой строке из второй колонки через пробел, а значения второй независимой переменной записываются из второй строки в первой колонке. Получается, что символ, находящийся на пересечении первой строки и первого столбца – пробел. На пересечении значений независимых переменных ставится значение зависимой переменной. Примером может служить таблица умножения Пифагора. Одномерные отображения позволяют задать до 8000 строк данных. Двухмерные отображения позволяют задать до 89x89 элементов данных (90x90 матрицу).

6. Nonlinear/max – блок выделения максимального сигнала или числа.

y = maximum (x_1, x_2).

Выход – наибольшее значение из входных величин. Этот блок не имеет параметров.

7. Nonlinear/merge – блок переключения входных сигналов.

Выходной сигнал принимает значение x_2, если $| x_1 | \geq 1$, и x_3 – в противном случае. Этот блок не имеет параметров.

8. Nonlinear/min – блок выделения минимального сигнала или числа.

y = minimum (x_1, x_3).

Выход – наименьшее значение входной величины. Этот блок не имеет параметров.

9. Nonlinear/quantize – блок округления или квантования по уровню.

Усекает или округляет входную величину до ближайшего числа, кратного разрешающей способности (точности), в зависимости от знака входа и разрешающей способности. Если вход и разрешающая способность имеют тот же самый знак, тогда сигнал усекается по величине. Если вход и разрешающая способность имеют противоположные знаки, то сигнал округляется по величине до следующего кратного разрешающей способности. Это лучше всего поясняется примером, приведенным в табл. 2.12.

Таблица 2.12

Округление или квантование по уровню

Вход	Выход	Разрешающая способность (точность, шаг)
1,9	1	1
–1,9	–1	–1
1,1	–1	2
–1,1	1	–2

Resolution определяет значение квантователя. По умолчанию – 0,05.

10. Nonlinear/relay – трехуровневое реле.

Моделирует блок-реле с тремя состояниями (–1, 0, 1). Параметр Dead Band показывает ширину зоны нечувствительности относительно нулевого значения входного сигнала и таким образом создается оператор-реле с тремя состояниями (–1, 0, 1). Зона нечувствительности должна быть положительна, ее значение по умолчанию – 0,2. Если зона нечувствительности равна 0, то моделируется реле с двумя состояниями (–1, 1).

11. Nonlinear/sampleHold – управляемый переключатель (включатель) фиксатор.

Выходной сигнал описывается следующим алгоритмом: если $|x_1| \geq 1$, то $y = x_2$, иначе y не изменяется. Блок фиксирует входную величину по управлению тактового сигнала. Параметр Initial Condition определяет начальное значение для y. По умолчанию – ноль.

2.8.1.6. Генераторы шумов (Random generators)

Описание блоков генераторов шумов (случайных чисел) – Random Generators.

1. Random Generators/gaussian – нормальный гауссовский шум.

Этот блок генерирует нормально распределенный шум. Параметр Mean показывает центр распределения шума. По умолчанию – ноль. Параметр

Standard Deviation показывает расстояние от среднего значения (среднее значение отклонения – девиация), которое занимает одна стандартная девиация. По умолчанию – единица. Начальное число случайной последовательности задается в диалоговом окне Simulate/Change Parameters…

2. Random Generators/uniform – произвольный однородный шум.

Этот блок однородно распределяет произвольный шум со значениями между 0 и 1. Параметр Time Delay (sec) определяет временную задержку перед вычислением величины шума. По умолчанию – ноль. Начальное число случайной последовательности задается в диалоговом окне Simulate/Change Parameters...

2.8.1.7. Блоки-задатчики времени (Real time)

1. Real time/rt-DataIn – блок считывания данных в реальном времени.

Считывает данные в реальном времени с аналогово-цифровой платы расширения компьютера. Заметим, что этот функциональный блок доступен только с приобретением расширения VISSIM/RT. Блок ввода данных в реальном времени содержит следующую информацию: заголовок; канал; разрешение канала; класс канала; тип канала.

2. Real time/rt-DataOut – блок записи данных через АЦП-плату в ПК.

Записывает данные в реальном времени на аналогово-цифровую плату расширения компьютера. Заметим, что этот функциональный блок доступен только с приобретением расширения VISSIM/RT. Блок ввода данных в реальном времени содержит следующую информацию: заголовок; канал; тип канала.

2.8.1.8. Блоки-получатели сигналов (Signal consumers)

Пакет VISSIM содержит следующие блоки-получатели (или регистраторы, или приемники) сигналов:

- CONSTRAINT – ограничение для статического уравнения;
- DISPLAY – цифровой вывод сигнала на экран;
- ERROR – флаг ошибки;
- EXPORT – экспорт, запись сигналов в файл данных;
- METER – вывод сигнала на стрелочное показывающее устройство типа вольтметра, изображенного на экране;
- PLOT – устройство вывода до четырех графиков выходных сигналов модели;
- STOP – блок условного останова моделирования.

1. Signal Consumers/constraint – блок указания точности решения алгебраического уравнения.

Ограничение для алгебраического уравнения используется вместе с «неопределенными» блоками. Раздел меню Tolerance определяет точность решения. Чем меньше допуск (ошибка), тем больше время вычисления. По умолчанию задан допуск 0,00001. Когда VISSIM обнаруживает блоки ограничения, то спрашивает, нужно ли активизировать итерацию Ньютона-Рафсона. Итерация Ньютона-Рафсона решает уравнение получением значений для «неопределенных» блоков, которые заставляют блоки ограничения стремиться к нулю насколько это возможно. Используйте команду Simulate/Change Parameters..., чтобы установить максимальное количество итераций, допустимую ошибку и возмущение для итерации Ньютона-Рафсона. Приложения включают неявные системы и подстройку системы.

2. Signal Consumers/display – блок цифрового указателя сигнала.

Отображает текущую величину входного сигнала с шестью или пятнадцатью значащими цифрами. По умолчанию – шесть значащих цифр. Используйте команду Edit/Preferences... для установки высокой точности отображения. Этот блок не имеет параметров.

3. Signal Consumers/error – блок остановки моделирования при $x \neq 0$.

Сообщает об ошибке в моделировании. Когда входной сигнал становится ненулевым, блок ошибки и все составные блоки, содержащие его, высвечиваются красным цветом и моделирование останавливается. Чтобы сбросить состояние ошибки, щелкните ПК мыши на блоке ошибки. Этот блок не имеет параметров.

4. Signal Consumers/export – блок записи данных в файл.

Записывает сигналы в файл данных формата ASCII. Используйте команду Edit/Add Input и Edit/ Remove Input для установки числа экспортируемых сигналов. По умолчанию – три, максимум – шестнадцать. Окно Data File Name определяет файл экспорта. Нажмите кнопку Open New File… для выбора файла из списка существующих файлов данных. Нажмите кнопку Browse Data... для чтения и редактирования выбранного файла данных. По умолчанию VISSIM записывает файл в ваш текущий каталог, используя имя файла диаграммы с расширением .DAT. Параметры окна Data Point Time Delta определяют после того, как VISSIM запишет временные интервалы информации в файл данных. По умолчанию – Fixed Interval. Выбирайте Fixed Interval, когда отметки данных расположены через фиксированные интервалы. Step Size по умолчанию определяется параметрами моделирования. Блок экспорта не осуществляет интерполяцию. Если вы определили интервал, отличный от размера шага моделирования, данные будут экспортироваться с интервалами, кратными целому числу шагов моделирования. Заметим, что эта автоматическая корректировка фактически невидима, потому что размер шага экспорта ни модифицируется в диалоговом окне экспорта, ни отражается в заголовке файла данных (см. ниже). Если вы импортируете этот файл данных в другой процесс моделирования, то должны изменить заголовок файла, чтобы отразить

истинный интервал экспорта. Блок импорта будет осуществлять интерполяцию, обеспечивающую сохранение этого выбора времени. Выберите Time Data Column, когда отметки данных происходят в нерегулярные временные интервалы. Некоторые прикладные программы не могут читать файлы данных без столбцов данных времени. В этом случае вы должны выбрать Time Data Column, даже если отметки данных происходят в фиксированных интервалах. Допустимые столбцы – от 1 до 16 включительно.

Окно Data File Info содержит следующую информацию: параметры Start Time и End Time представляют диапазон времени для записываемых данных. Они только для чтения, получаются из параметров моделирования. Окно Data Point Count определяет максимальное число данных, записываемых в файл данных. Каждая отметка данных занимает 8 байтов памяти. Максимально возможное число элементов данных – 128000. По умолчанию – 512 элементов данных, что требует 4096 байтов памяти на каждый вход. Если выбран параметр Periodic Data Flush, то буфер экспорта будет отключен на промежутке времени, определяемом пользователем и устанавливаемом в окне Flush Interval. Параметр Suppress VisSim Header полезен в случае, если файл данных необходимо импортировать в программу другого типа, отличного от VISSIM. Информация заголовка определяется из значений в диалоговом окне экспорта. В нем записываются данные о временных интервалах для файла с использованием следующего формата: Fixed Interval #I=start time, end time, increment Variable Interval #T=number time column. Параметр Append To File позволяет добавить экспортируемые данные к существующим файлам при каждом выполнении моделирования, вместо того чтобы перезаписывать файл в начале каждого нового выполнения. Это полезно для многократно выполняемых приложений типа сбора данных, моделирования Монте Карло и обучаемой нейросети. Параметр Digits of Precision указывает значение точности для экспортируемых данных. По умолчанию – 15 значащих цифр.

5. Signal Consumers/meter – стрелочный измеритель сигнала типа вольтметра.

Масштабирование и промежуточная градуировка выбираются автоматически, исходя из установленных в меню верхней и нижней границ. Lower Bound – наименьшая величина, отображаемая прибором. По умолчанию –(–10). Upper Bound – наибольшая величина, отображаемая прибором. По умолчанию – 10.

6. Signal Consumers/plot – графопостроитель регистрируемых сигналов.

Одновременно рисует графики до четырех входных сигналов в двумерной координатной сетке. Используйте команды Edit/Add Input и Edit/Remove Input, чтобы изменить число входных меток на графическом блоке. Не имеется никаких ограничений на число или расположение графических блоков в вашей схеме.

Щелкните ПК мыши на графическом блоке для обращения к Parameter Dialog Box (меню блока), который содержит все параметры, доступные вам.

Параметр Fixed Bounds определяет, изменяет ли VISSIM границы графической шкалы для отображения входного сигнала, выходящего за границы диапазона, или отсекает значения сигнала вне существующих графических пределов. По умолчанию этот параметр выключен, поэтому весь график обновляется каждый раз, когда необходимо изменить масштаб. Если вы завершаете моделирование с этим отключенным параметром, VISSIM сам установит пределы. Вы можете с помощью Fixed Bounds ускорять последующие выполнения. Fixed bounds позволяет Вам устанавливать верхнюю и нижнюю границы *X* и *Y*. Чтобы исследовать часть существующего графика, введите поддиапазон, представляющий интерес, и эта часть графика будет отображена в полной графической области. Параметры моделирования имеют старшинство над пределами графика по координате *x*.

Параметр Frequency Domain обеспечивает частотный энергетический спектр, используя быстрое преобразование Фурье (*FFT* алгоритм). Если ваш график *FFT* имеет непредвиденные скачки, проверьте адекватность частоты выборки для получения точных результатов в диалоговом окне Simulate/Change Parameters... Основываясь на размере шага и диапазоне, установите Max Plotted Points так, чтобы вы действительно рисовали график на каждом шаге. Параметр Point Plot определяет, выводится ли график как ряд точек или как непрерывная линия. По умолчанию этот параметр выключен, что создает график в виде линии.

Параметр Max Plotted Points определяет точность и гладкость графика. При составлении графика нескольких сигналов на черно-белом дисплее или принтере вы можете уменьшить количество точек, увеличивая расстояние между ними, но сохраняя четкое изображение линий. Каждая координата данных потребляет 8 байтов памяти. Максимальное количество используемых точек графика – 128000 на входной сигнал. По умолчанию – 512, что требует 4096 байтов памяти на каждый вход.

Параметр Over Plot определяет, уничтожается ли предыдущий график или сохраняется. Используйте его для сравнения вариантов из серии моделирования. По умолчанию этот параметр выключен, поэтому каждое моделирование начинается с новым графическим экраном. Включенный параметр Over Plot расходует память.

Параметр Plot Count определяет разрешенное количество последовательно выводимых графиков. По умолчанию – четыре, что требует четыре раза по 4096 байтов памяти для каждого входа.

Параметр Geometric Markers позволяет идентифицировать сигналы с использованием квадратиков, ромбов, кругов и треугольников для графиков каждого сигнала. По умолчанию Geometric Markers не используются.

Параметр Marker Count определяет число маркеров для различных графиков. По умолчанию – десять.

Параметр Grid Lines определяет, появляется ли координатная сетка на графиках. Координатная сетка помогает определять графические координаты. По умолчанию Grid Lines включены.

Параметры Log *X* и Log *Y* определяют, будет ли ваш график линейный, логарифмический или полулогарифмический: по умолчанию устанавливается линейный. Для его получения не выбирайте никакого параметра. Для получения логарифмического выберите оба параметра. С целью получения полулогарифмического графика выберите или Log *X*, или Log *Y*. Заметим, что вы не можете изображать отрицательные величины на логарифмической оси. Любое отрицательное значение будет отсекаться за нижний предел шкалы.

Параметр *XY* Plot определяет, отображаются ли графики в режиме *XY* или во временной области независимо от масштабирования оси. Когда действует *XY* Plot, верхний сигнал представляет *x*-ось, а нижний – *y*-ось. Отключите *XY* Plot, чтобы установить режим временной области. Тогда VISSIM будет рисовать все сигналы на *y*-оси, а время на *x*-оси. По умолчанию этот параметр выключен.

Параметр *X* Axis позволяет вам определять входной сигнал, который нужно использовать для *x*-оси. Когда этот параметр выключен, по умолчанию им считается первый из четырех входных сигналов на графическом блоке.

Параметр Title устанавливает заголовок графика до 80 символов. Графики первоначально идут без заглавия. Параметр Subtitle устанавливает подзаголовок графика до 80 символов только для копирования на принтер. Первоначально графики не имеют подзаголовка.

Параметры *X* Label и *Y* Label устанавливают метки оси до 80 символов. По умолчанию *y*-ось не помечена. Метка по умолчанию для *x*-оси – время в секундах. В *XY* графике метка для *x*-оси является маркировкой входного сигнала, устанавливаемой с помощью кнопки Label Signals..., которая вызывает диалоговое окно, куда вы можете вводить обозначение до 80 символов для каждого входного сигнала. Обозначения появляются в верхнем левом углу графика, с шаблоном линии, используемой для графика каждого сигнала. На цветном дисплее или устройстве печати метки имеют тот же цвет, что и соответствующие сигналы. Все входные сигналы первоначально не обозначены. Когда вы выбираете сигнал для *x*-оси *XY* графика, обозначение этого сигнала автоматически становится меткой для *x*-оси.

Параметр Read Coordinates позволяет вам получать численные значения графических координат. Курсор мыши становится крестиком по размеру графика внутри графической области, при этом в левом нижнем углу будет появляться числовое значение координаты. Щелкните ЛК или ПК мыши, чтобы выйти из режима Read Coordinates. Для перемещения графического блока установите курсор мыши внутри графического экрана и установите его на новое место.

Чтобы изменять размеры графического блока, используйте кнопки максимизации или минимизации, расположенные в верхнем правом углу

графического блока, или нажмите ЛК мыши на границе графического блока и установите желаемый размер. Для получения печатной копии одиночного графика щелкните на пункте меню Control графического блока и выберите команду Print. VISSIM по умолчанию поставит график в очередь на системный принтер. VISSIM всегда выводит графики с подгонкой до полной страницы и не включает входные соединительные метки графического блока, минимизируемый/максимизируемый блок, или пункт меню Control.

7. Signal Consumers/stop – блок остановки моделирования при $x \geq 1$.

Если $x \geq 2$, то происходит остановка текущего выполнения. Если режим автоперезапуска, то не производится следующий запуск. Иначе, если $x \geq 1$, тогда произойдет остановка текущего выполнения. Если режим автоперезапуска, то моделирование начинается сначала. Иначе, как обычно, останавливает моделирование, когда входной сигнал больше или равен единице. Этот блок не имеет параметров.

2.8.1.9. Блоки задания сигналов (Signal producer)

Пакет VISSIM содержит следующие блоки-генераторы сигналов:

- BUTTON – кнопка;
- CONST – константа, блок задания постоянного сигнала;
- IMPORT – импорт (чтение) данных из входного файла;
- PARABOLA – параболический сигнал;
- PULSETRAIN – периодическая последовательность нулей и единиц;
- RAMP – линейно нарастающий сигнал;
- REALTIME – реальное время в миллисекундах.
- SINUSOID – синусоидальная функция;
- SLIDER – управляемый мышью аналоговый сигнал, подобие потенциометра;
- STEP – функция-скачок;
- UNKNOWN – блок неизвестных, используется для решения алгебраических уравнений в неявных системах.

Блоки (почти все) имеют только один выход.

1. Signal Producers/button – кнопка для генерации скачка $1(t)$.

Если подвести курсор и нажать на кнопку, то $y = 1$ [кнопка темная], если нажать еще раз, то $y = 0$ [кнопка светлая]. Позволяет вам динамически вставлять 0 и 1 во время моделирования. Переключение цвета блока осуществляется щелчком ПК мыши на блоке. Имя кнопки изменяется командой Edit/Rename Block.... Этот блок не имеет параметров.

2. Signal Producers/const – блок генерации постоянного числа (сигнала).

y = value.

Генерирует постоянный сигнал. Value показывает значение величины (по умолчанию – один) выходного сигнала. Возможно использование π и ее констант. Величина сигнала устанавливается пользователем (после нажатия ПК мыши).

3. Signal Producers/import – блок воспроизводства данных из файла в виде непрерывного сигнала.

Записывает сигналы из ASCII файла данных. Используйте команды Edit/Add Input и Edit/ Remove Input для установки числа выходных сигналов. По умолчанию – три, максимум – шестнадцать. Массиву данных могут быть установлены начало, конец и значения приращения таким образом, чтобы импортируемые значения соотносились со временем моделирования. Значения данных будут линейно интерполироваться, если шаг по времени массива данных отличается от шага моделирования. Окно Data File Name определяет имя файла для импорта. Нажмите кнопку Open New File... для выбора файла из списка существующих файлов данных. Нажмите кнопку Browse Data... для чтения и редактирования выбранного файла данных. Окно Data Point Time Delta указывает интервал времени между отметками данных в файле. Если файл данных – файл пакета VISSIM, то автоматически считывается информация о интервале времени из заголовка файла и соответственно устанавливаются параметры. По умолчанию – Fixed Interval. Если отметки данных находятся в фиксированных интервалах, то выбирайте Fixed Interval, иначе введите интервал в соответствующее текстовое окно. Выберите Time Data Column, когда отметки данных происходят в нерегулярные временные интервалы. Введите номер столбца, содержащего отметки данных времени, в соответствующее текстовое окно. Допустимые номера столбцов – от одного до шестнадцати. Окно Data File Info содержит разделы только для чтения Start Time и End Time, представляющие диапазон времени, в который записываются данные и, следовательно, диапазон времени, на котором допустимо моделирование. Эта информация получается из заголовка файла. Информация из заголовка сообщает VISSIM интервалы времени для данных файла в следующем формате: Fixed Interval #I=start time, End time, Increment Variable Interval #T=numberитime column. Параметр Data Point Count определяет максимальное число элементов данных для считывания в VISSIM. VISSIM будет извлекать эту информацию из файла непосредственно. Максимально допустимое число элементов данных – 128000. Каждый элемент данных потребляет 8 байтов памяти на столбец.

4. Signal Producers/parabola – блок квадратичного сигнала $y = a(t - c)^2$.

Создает параболический сигнал исходя из времени моделирования. Параметр Time Delay (c) определяет величину задержки в секундах (по умолчанию – ноль) перед вычислением значения выходного сигнала. Параметр Slope Rate (a) масштабирует кривизну параболы. Значение по умолчанию – один.

5. Signal Producers/pulseTrain – блок-генератор импульсов.

Генерирует последовательность импульсов единичной амплитуды. Это полезно для синхронизации unitDelays и sampleHolds. Параметр Time Delay (секунд) определяет величину задержки перед вычислением значения выходного сигнала. Значение по умолчанию – 0. Параметр Time Between Pulses определяет время между импульсами. По умолчанию – 0,01. Вы можете добавить два входа к блоку pulseTrain, используя команду Edit/Add Input. Первый дополнительный вход позволяет вам вводить запаздывание извне, а второй – вводить извне время между импульсами. Эти дополнительные входы отменяют существующие параметры.

6. Signal Producers/ramp – блок линейного сигнала $y = a\,(\,t - t_{\text{задержки}}\,)$.

y = slope · (time – time delay)= $a\,(\,t - t_{\text{задержки}}\,)$,

y = фронт сигнала · (текущее время – время задержки).

Создает единичный пилообразный сигнал, исходя из времени моделирования. Параметр Time Delay ($t_{\text{задержки}}$) – сдвиг во времени в секундах при вычислении значения выходного сигнала (по умолчанию – ноль). Параметр Slope (a) – фронт сигнала, по умолчанию – единица.

7. Signal Producers/realTime – блок источника времени ^{bt} ($b = 10^{-3}$).

Выдает время в миллисекундах с начала вашего текущего сеанса VISSIM. Этот блок не имеет параметров. Обратите внимание, что это не время моделирования. Для получения времени моделирования выберите блок пилообразного сигнала.

8. Signal Producers/sinusoid – генератор синусоиды $y = a\sin[\omega(t - \tau)]$.

Создает синусоидальный сигнал исходя из времени моделирования. Параметр Time Delay (τ) – сдвиг во времени в секундах при вычислении значения выходного сигнала. Его значение по умолчанию – ноль. Параметр Frequency (ω) определяет частоту выходного синусоидального сигнала, определяется в рад/с. Значение по умолчанию – единица, что соответствует синусоиде с периодом 2π секунды. Параметр Amplitude (a) определяет максимальную величину выходного сигнала. Значение по умолчанию – единица.

9. Signal Producers/slider – блок скользящего регулятора сигнала типа потенциометра.

Позволяет с помощью мыши динамически изменять значение сигнала во время моделирования. Блок сдвига отображает текущую величину сигнала. Используйте горизонтальную линейку в блоке сдвига, чтобы корректировать величину сигнала между верхним и нижним пределом. Параметр Current Value определяет начальное значение выхода (по умолчанию – ноль). Параметр Upper Bound определяет максимальное значение выхода (по умолчанию – 100).

Параметр Lower Bound определяет минимальное значение выхода (по умолчанию – 100).

10. Signal Producers/step – блок источника сигнала типа скачка $y = a_1(t - \tau)$.

Если время меньше времени запаздывания Time delay (τ), то $y = 0$, иначе $y = a$. Создает единичный ступенчатый сигнал исходя из времени моделирования. Параметр Amplitude (a) определяет максимальную величину выходного сигнала. Значение по умолчанию – единица. Параметр Time Delay (τ) – сдвиг во времени при вычислении значения выходного сигнала в секундах. Значение по умолчанию – ноль.

11. Signal Producers/unknown – блок начального задания при решениях неявных алгебраических уравнений.

Применяется в неявных системах и в подстройке системы. «Неопределенности» должны быть всегда присоединены непосредственно или косвенно к блокам ограничения. VISSIM решает уравнение при численном воздействии на неопределенности, чтобы устремить ограничения к нулю. Максимальное количество итераций, допустимая ошибка и возмущение устанавливаются в команде Simulate/ Change Parameters... Этот блок не имеет параметров.

2.8.1.10. Блоки задержки (Time delay)

1. Time Delay/Time Delay – блок динамической задержки.

$y = x_2 e^{-x_1 s}$.

Осуществляет задержку x_2 в течение времени, определяемого x_1, где s – оператор Лапласа. Параметр Initial Condition устанавливает начальное условие для y. По умолчанию - ноль. Параметр Max Buffer Size управляет максимальным размером внутреннего буфера. Блоку timeDelay требуется элемент буфера для каждого шага во время необходимой задержки. Если timeDelay требует больше буферных элементов, чем позволяет Max Buffer Size, отметки данных будут потеряны, порождая прерывистость, возникающую в результате задержки выходного сигнала. Если сигнал – прерывистый, увеличьте величину Max Buffer Size. Для вычисления размера буфера определите максимальную требуемую задержку, разделите на шаг моделирования и добавьте 1. По умолчанию – 128 буферных элементов, а максимум – 8191. Каждый элемент занимает 8 байтов памяти.

2. Time Delay/unitDelay – блок синхронизированной постоянной задержки.

Если $|x_1| \geq 1$, то y = ybuffer, ybuffer = x_2, иначе y и ybuffer не изменяются. Определяет синхронизированное запаздывание для входных сигналов. Этот блок задает цифровую задержку в модели при непрерывном моделировании, обычно создается присоединением блока Signal Producers/pulseTrain к

соединительной метке x_1 блока unitDelay. Параметр Initial Condition устанавливает начальное значение для *y* (по умолчанию равное нулю).

2.8.1.11. Блоки трансцендентных функций (Transcendental)

Пакет VISSIM содержит следующие блоки трансцендентных функций с одним входом и одним выходом:

ACOS – блок вычисления арккосинуса;

ASIN – блок вычисления арксинуса;

ATAN2 – блок вычисления арктангенса;

BESSEL – блок вычисления функции Бесселя;

COS – блок вычисления косинуса;

COSH – блок вычисления гиперболического косинуса;

EXP – блок вычисления экспоненты;

LN – блок вычисления натурального логарифма;

LOG10 – блок вычисления десятичного логарифма;

SIN – блок вычисления синуса (блок-генератор синусоиды – SINUSOID, см. п. 2.8.1.9);

SINH – блок вычисления гиперболического синуса;

SQRT – блок вычисления квадратного корня;

TAN – блок вычисления тангенса;

TANH – блок вычисления гиперболического тангенса.

Блоки осуществляют функциональное преобразование сигналов в пределах допустимых значений для данных функций входных сигналов.

2.8.1.12. Блок собственной функции-программы (Userfunction)

Вызывает DLL функцию из вашей диаграммы. Для пользователей VISSIM в основном каталоге VISSIM установлен файл README.TXT, содержащий описание работы с DLL функциями. Эти файлы могут быть скомпилированы с использованием Microsoft C, Microsoft Fortran, Borland C++. Ограничения: 255 входов, 16 выходов, 12 описательных параметров и сколько угодно произвольных параметров.

2.8.1.13. Составные блоки (Edit/Greate Compound Block)

Составные блоки заменяют собой группу блоков, связанную в один сложный блок. Составные блоки аналогичны подпрограммам (функциям) других языков программирования и обеспечивают модульный принцип разработки проектов. Составные блоки могут создаваться, копироваться, сохраняться, как файлы, и удаляться, как обычные стандартные блоки системы.

Ограничения числа составных блоков или уровней формирования схем нет. Нажмите ПК мыши на составном блоке для просмотра его структуры. Для возврата на верхний уровень нажмите ПК мыши на свободном поле экрана.

Чтобы создать составной блок, сначала выберите блоки, которые вы хотите сгруппировать (инкапсулировать). Установите курсор в левом верхнем углу области экрана, которую занимают группируемые блоки, и нажмите ЛК мыши. Удерживая клавишу, смещайте курсор в противоположный угол области, при этом на экране образуется прямоугольный контур, который должен охватить выбираемые блоки. После отпускания клавиши блоки, содержащиеся в контуре, будут выбраны (выделяются красным цветом). Выберите команду меню Edit/Create Compound Block, при этом появится окно диалога, куда вы можете ввести имя составного блока. Имя должно содержать не больше чем 32 символа и не должно содержать точечный символ (.). Система перерисовывает схему с отображением созданного составного блока, присоединяет метки табуляции соединителей и автоматически создает внутренние соединения для всех несвязанных меток табуляции соединителей на внутренних блоках (кроме входных соединителей на глобальных переменных) и для всех меток табуляции соединителей к внешним блокам. Для просмотра составного блока нажмите ПК мыши на блоке.

Для отмены инкапсуляции необходимо, войдя внутрь сложного блока, скопировать или переместить схему блока в буфер системы командой Edit/copy или Edit/cut, затем, выйдя на верхний уровень иерархии, удалить сложный блок и установить содержимое сложного блока из буфера командой Edit/paste. При этом все внешние соединения установленной части схемы необходимо восстановить «вручную».

2.8.2. Задания для самостоятельной работы

1. В рабочей области пакета VISSIM собрать схему одноконтурной САР, (см. рис. 2.30). Модель ОУ описывается упрощенной формой ПФ ОУ (2.8). Модель регулятора описывается также ПФ динамических звеньев для П-регулятора (2.10), ПИ-регулятора (2.11). Передаточная функция ПИД-регулятора имеет следующий вид:

$$R(s) = k_{\text{р}} + \frac{k_{\text{р}}}{T_{\text{и}} s} + \frac{T_{\text{д}} s}{0{,}25 T_{\text{д}} s + 1}, \qquad (2.61)$$

где $k_{\text{р}}$ – коэффициент передачи регулятора в процентах хода регулирующего органа на единицу регулируемой величины (так как выходная величина регулятора – ход исполнительного механизма – обычно равна входной величине ОУ – ходу регулирующего органа); $T_{\text{и}}$ – постоянная интегрирования (время изодрома), с; $T_{\text{д}}$ – постоянная дифференцирования (время предварения), с.

Собирая схему ОУ, используйте блоки TransferFunction, Time Delay, CONST. Числовые значения ПФ ОУ $k_{об}$, $T_{об}$ и $\tau_{об}$ выбрать из табл. 2.13, номер варианта уточнить у преподавателя.

Таблица 2.13

Варианты задания

Номер вари-анта	$k_{об}$	$T_{об}$	$\tau_{об}$	Регу-лятор	$k_р^о$	$T_И^о$	$T_Д^о$
1	2,0	8	2,4	П ПИ ПИД	1,473 1,325 1,767	– 7,211 4,300	– – 1,081
2	3,0	10	3,0	П ПИ ПИД	0,982 0,883 1,178	– 9,014 5,384	– – 1,352
3	5,0	12	3,6	П ПИ ПИД	0,589 0,530 0,707	– 10,817 6,460	– – 1,623
4	6,0	14	4,2	П ПИ ПИД	0,491 0,442 0,589	– 12,620 7,537	– – 1,893
5	2,0	8	4,0	П ПИ ПИД	0,952 0,856 1,142	– 11,396 6,806	– – 1,709
6	0,2	4	2,0	П ПИ ПИД	9,517 8,566 11,421	– 5,698 3,403	– – 0,855
7	0,3	5	2,5	П ПИ ПИД	6,345 5,710 7,614	– 7,123 4,254	– – 1,068
8	0,5	6	3,0	П ПИ ПИД	3,807 3,426 4,568	– 8,547 5,104	– – 1,282
9	2,0	100	30,0	П ПИ ПИД	1,473 1,325 1,767	– 90,144 53,836	– – 13,523
10	3,0	120	36,0	П ПИ ПИД	0,982 0,884 1,178	– 108,17 64,604	– – 16,228

Собирая схему регулятора, используйте блоки GAIN, SummingJunction, TransferFunction, как параллельные звенья П, ПИ и ПИД-регуляторов. Базовые (оптимальные) настройки регуляторов $k_р^о$, $T_И^о$, $T_Д^о$, определенные методом Циглера-Никольса (см. п. 2.5.2.5), также выбрать из табл. 2.13.

2. Для анализа качества САР во временной области необходимо на вход схемы подать единичный скачок STEP, к выходу подключить блок PLOT. Базовый вариант эксперимента зафиксировать вместе со схемой. Пример схемы для САР с П-регулятором представлен на рис. 2.93.

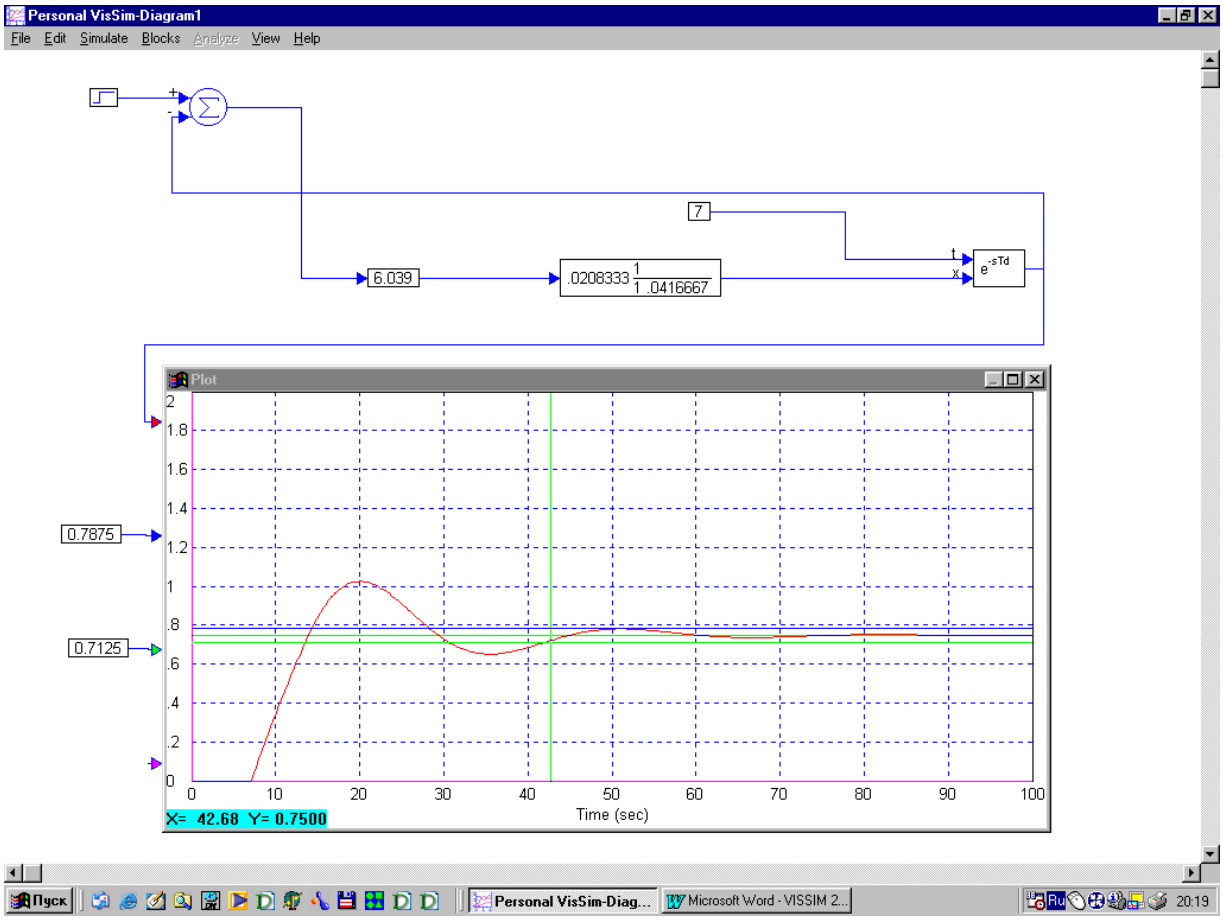


Рис. 2.93. Схема САР с П-регулятором

3. С целью определения устойчивости системы провести полный факторный эксперимент (ПФЭ) типа 2^k [2.9]. В качестве факторов эксперимента выбрать $k_{об}$, $T_{об}$ и $\tau_{об}$. Результаты ПФЭ представить в виде семейства переходных характеристик (по 4 графика при $k_{об} + \Delta k_{об}$ и $k_{об} - \Delta k_{об}$), используя флаг OVERPLOT в блоке PLOT (рис. 2.94). В случае неустойчивого характера системы результат зафиксировать отдельно (рис. 2.95). ПФЭ провести для САР с П, ПИ, ПИД-регуляторами отдельно.

4. По переходным характеристикам найти показатели качества САР: время регулирования $t_р$, максимальное динамическое отклонение σ, перерегулирование η для каждой реализации. На рис. 2.96 представлены показатели качества САР, получаемые согласно следующим формулам.

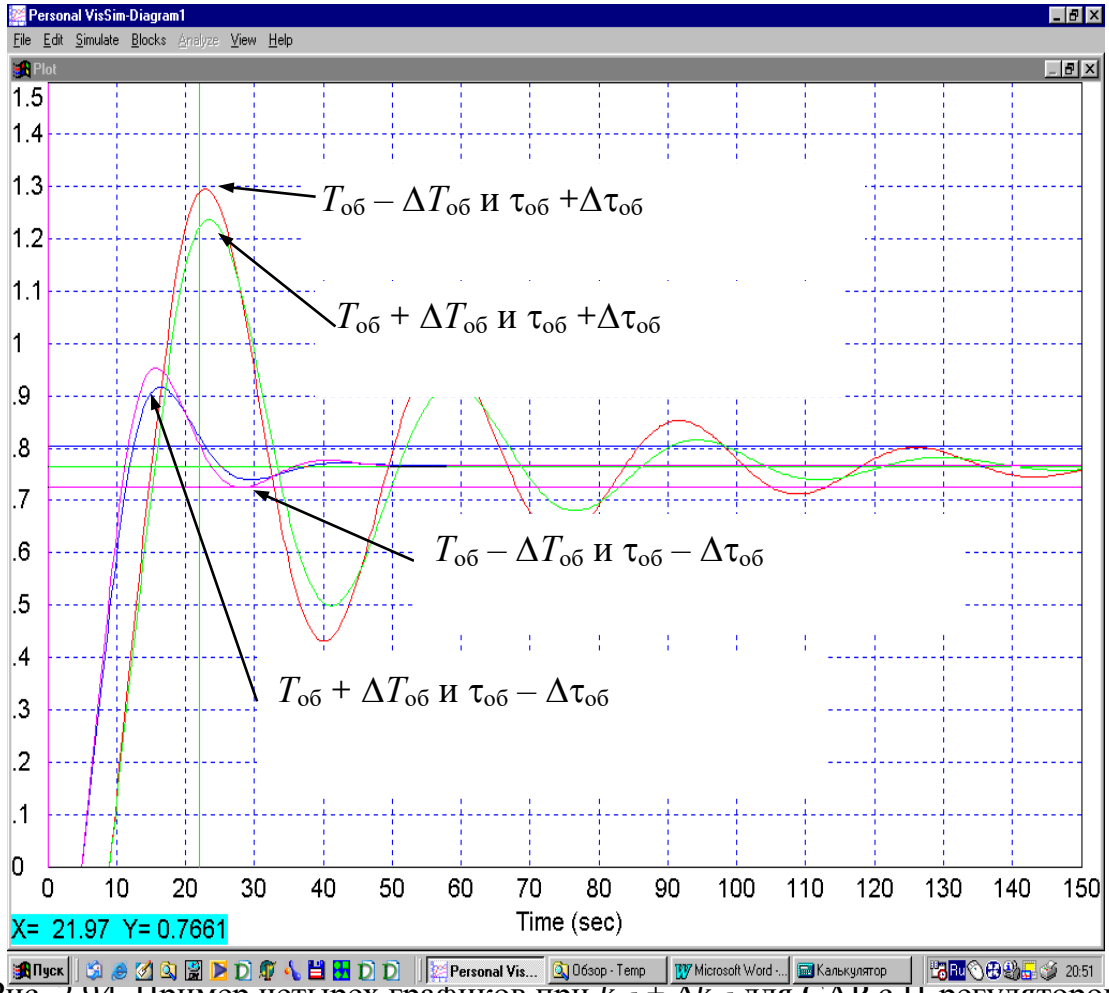


Рис. 2.94. Пример четырех графиков при $k_{об} + \Delta k_{об}$ для САР с П-регулятором

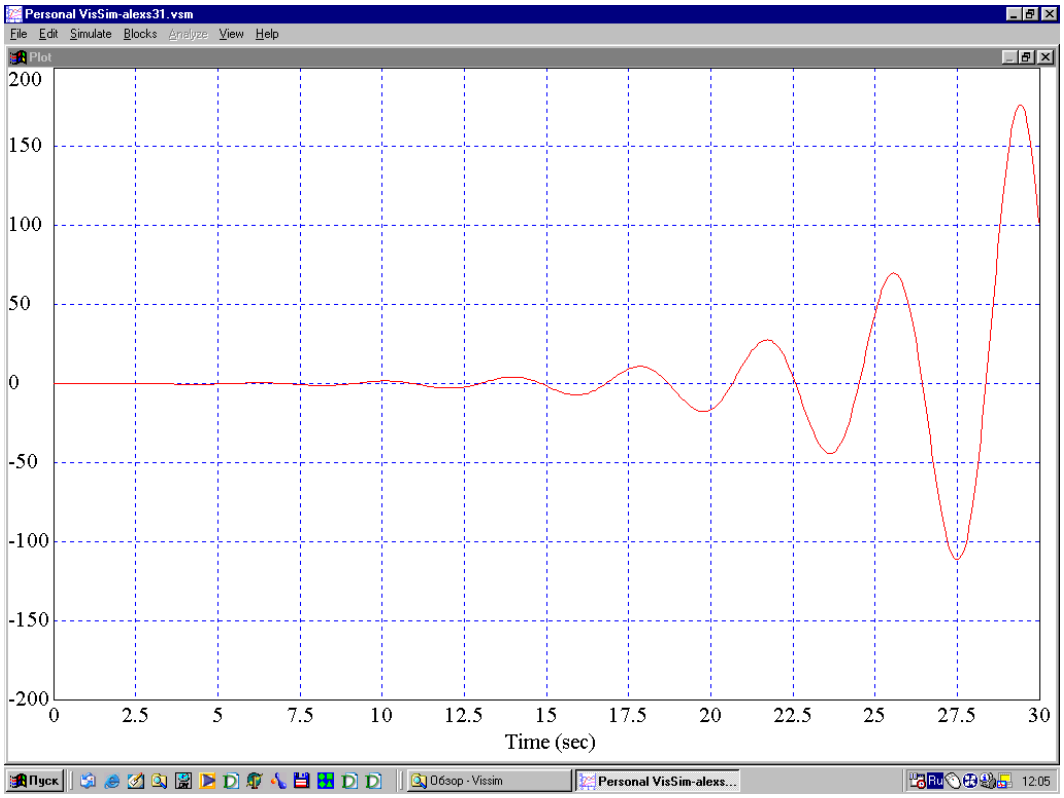


Рис. 2.95. Пример неустойчивого характера САР

Время регулирования $t_р$ в секундах определяется при условии, что сигнал зашел и больше не выходит из δ-трубки, где δ = 5 %.

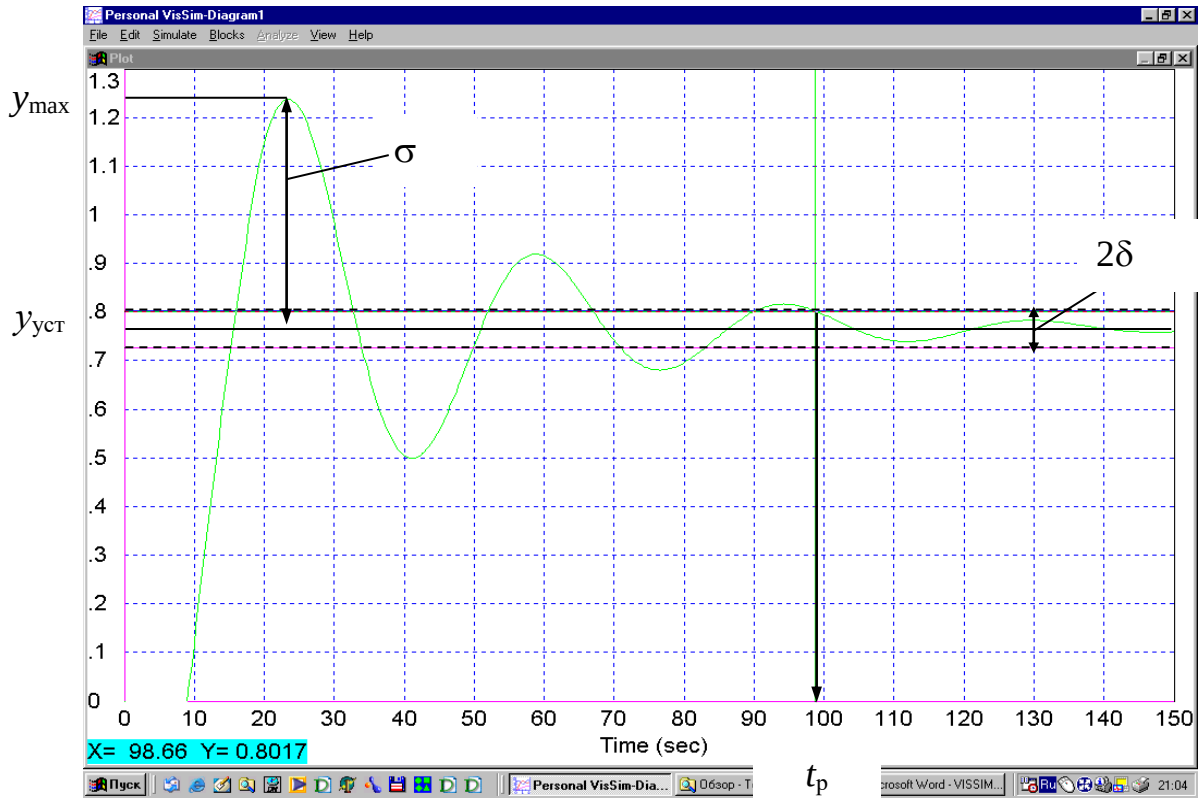


Рис. 2.96. Пример определения показателей качества САР с П-регулятором по переходной характеристике

Максимальное динамическое отклонение по управлению в единицах регулируемой величины

$$\sigma = y_{max} - y_{уст}. \qquad (2.62)$$

Перерегулирование в процентах по управлению

$$\eta = 100\ (y_{max} - y_{уст}) / y_{уст}. \qquad (2.63)$$

Результаты представить в виде обобщенной таблицы. Выявить закономерности влияния варьируемых параметров на показатели качества системы.

5. Результаты моделирования САР представить следующим образом:

А. Полученную в рабочей области VISSIM модель одноконтурной САР с конкретными данными ОУ ($k_{об}$, $T_{об}$ и $\tau_{об}$), базовыми настройками регуляторов ($k_р^о$, $T_И^о$, $T_Д^о$), а также базовые выходные параметры переходного процесса (время регулирования, максимальное динамическое отклонение, перерегулирование САР).

Б. Результаты ПФЭ с моделью системы: семейство переходных характеристик для П, ПИ, ПИД-регуляторов отдельно, обобщенную таблицу

результатов (время регулирования, максимальное динамическое отклонение, перерегулирование СУ).

В. Выявить и написать выводы о закономерности влияния варьируемых параметров системы ($k_{об}$, $T_{об}$ и $\tau_{об}$) на показатели качества системы (время регулирования, максимальное динамическое отклонение, перерегулирование).

2.9. МОДЕЛИРОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ С ПОМОЩЬЮ MATLAB

2.9.1. Стандартные блоки пакета программ MATLAB

Пакет программ MATLAB [2.7] предназначен для решения большого круга задач, в том числе для моделирования процессов в системах управления объектами, которые можно описать дифференциальными уравнениями или передаточными функциями Ниже более подробно описаны функций блоков, которые необходимы для моделирования САР.

2.9.1.1. Раздел Continuous

Раздел Continuous (непрерывные системы) содержит блоки, которые можно условно разделить на две группы: блоки, непосредственно предназначенные для описания непрерывных систем, и блоки общего назначения, которые могут быть использованы в модели любой системы.

1. *Transfer Fcn* (передаточная функция)

С помощью блока Transfer Fcn можно задать передаточную функцию динамического звена. Для этого существуют два поля – Numerator и Denominator (рис. 2.97).

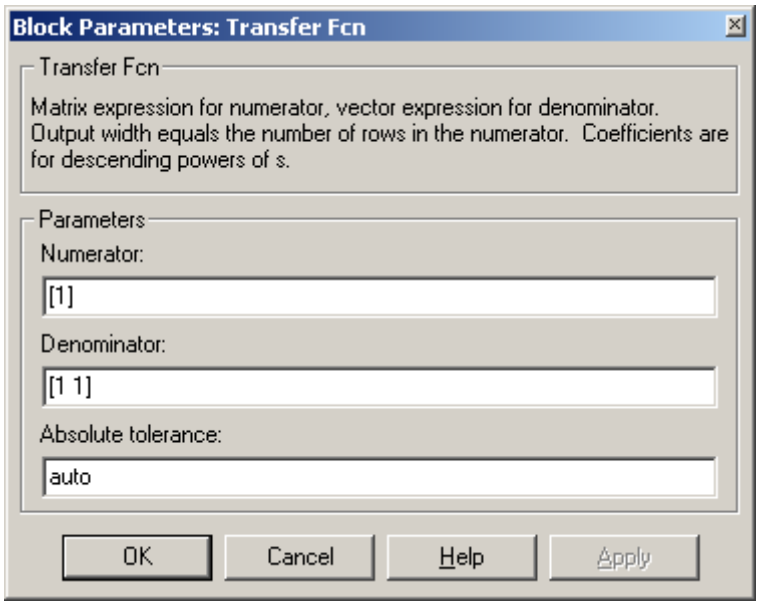


Рис. 2.97. Окно настройки блока Transfer Fcn

Поле Numerator служит для ввода коэффициентов многочлена числителя. Поле Denominator – для ввода коэффициентов многочлена знаменателя.

Коэффициенты числителя и знаменателя вводятся через пробел, начиная с самого высокого порядка. Знаки препинания или *s*-операторы не включаются. MATLAB определяет порядок передаточной функции по числу коэффициентов знаменателя. Например, передаточная функция *n*-го порядка будет иметь *n* + 1 коэффициентов знаменателя.

2. *Transport* Delay (блок транспортного запаздывания)

Данный блок реализует произвольную задержку передаваемого сигнала. Настройка блока производится с помощью четырех параметров (рис. 2.98):

- Time delay (время задержки) – количество шагов модельного времени, на которое задерживается сигнал. Может вводиться либо в числовой форме, либо в форме вычисляемого выражения;
- Initial input (начальное значение на входе) – значение амплитуды входного сигнала в момент инициализации блока (по умолчанию равно 0);
- Initial buffer size (начальный размер буфера) – объем памяти (в байтах), выделяемой в рабочей области MATLAB для хранения задержанного сигнала. Значение параметра должно быть кратно 8 (по умолчанию – 1024);
- Pade order (порядок линеаризации Паде) – в данном поле задается порядок линеаризации (по умолчанию поле равно 0).

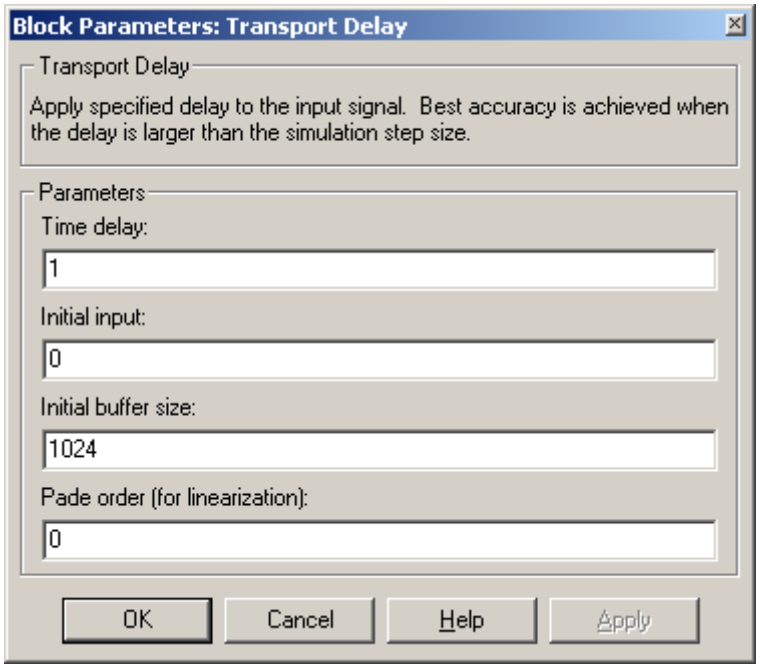


Рис. 2.98. Окно настройки блока Transport Delay

2.9.1.2. Раздел Math

Раздел Math (математические блоки) содержит блоки, которые реализуют элементарные (алгебраические и тригонометрические) функции, а также операции математической логики и могут быть использованы в модели любой системы.

1. *Блок Gain* (умножитель)

Блок Gain выполняет роль умножителя сигнала, поступающего на его вход. Блок имеет два параметра настройки (рис. 2.99):

- в поле Gain (множитель) можно ввести числовую константу, переменную или вычисляемое выражение;
- флажок Saturate on integer overflow (подавлять переполнение для целых) дает возможность задать необходимость «урезания» результата сложения, если он превышает диапазон, установленный для целочисленных значений.

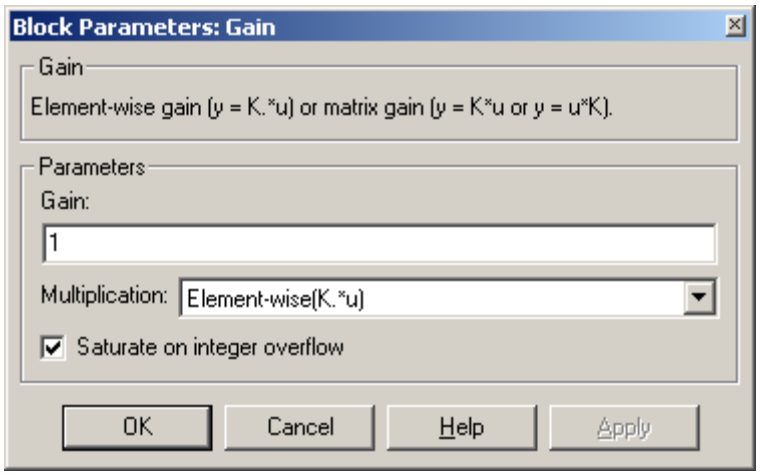


Рис. 2.99. Окно настройки блока Gain

2. *Блок Sum* (сложение)

Блок Sum выполняет суммирование входных сигналов. Он может использоваться в двух режимах:

- сложение входных сигналов (в том числе с разными знаками);
- суммирование элементов вектора, поступающего на вход блока. Вводя значения в поле List of Signs (список знаков), можно управлять режимами работы блока (рис. 2.100).

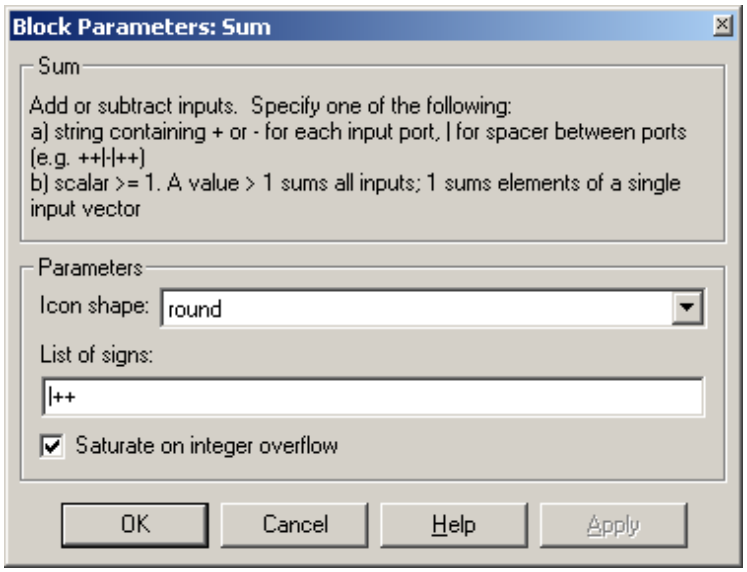


Рис. 2.100. Окно настройки блока Sum

Значения могут задаваться одним из трех способов:

- в виде последовательности знаков + и –, причем число знаков определяет число входов блока, а сами знаки – полярности соответствующих входных сигналов; при большом количестве слагаемых их целесообразно разбивать на несколько групп, отделяя одну группу от другой символом | (например: ++| – –) без пробелов;
- в виде целой положительной константы (большей 1), значение которой равно числу входов блока, а все входы считаются положительными (например, ввод константы 4 аналогичен вводу «списка знаков» в форме ++++);
- ввод значения 1 соответствует вычислению суммы элементов входного вектора (в этом случае внутри блока выводится символ ?).

Два других параметра настройки блока имеют следующий смысл:

- раскрывающийся список Icon shape (форма значка) позволяет выбрать форму блока: round (окружность) или rectangular (прямоугольник);
- флажок Saturate on integer overflow (подавлять переполнение для целых) дает возможность задать необходимость «урезания» результата сложения, если он превышает диапазон, установленный для целочисленных значений.

2.9.1.3. Раздел Sinks

В данном разделе размещены блоки-получатели сигналов. Условно их можно разделить на три вида:

1. Блоки, используемые при моделировании в качестве «смотровых окон»:

- блок Scope (осциллограф);
- блок XYGraph (двумерный график) – обеспечивает создание двумерных графиков в прямоугольной системе координат;
- блок Display (экран) – предназначен для отображения численных значений величин.

2. Блоки, обеспечивающие сохранение промежуточных и/или выходных результатов моделирования:

- блок To File (запись в файл);
- блок To Workspace (запись в рабочую область);

3. Блок управления моделированием – Stop Simulation (остановка моделирования) – позволяет прервать моделирование при выполнении тех или иных условий. Блок срабатывает в том случае, если на его вход поступает ненулевой сигнал.

Рассмотрим подробно блок Scope. Блок Scope позволяет в процессе моделирования наблюдать динамику интересующих исследователя

характеристик системы. Создаваемое с его помощью «смотровое окно» напоминает экран измерительного прибора. Размер и пропорции окна можно изменять произвольно, перемещая его границы мышью. Одновременно в окне Scope может отображаться до 30 кривых.

Для управления параметрами окна Scope в нем имеется панель инструментов, содержащая следующие функции (рис. 2.101):

- Zoom (масштаб) – изменение масштаба осей графика;
- Zoom *X*-asis (масштаб по оси *X*) – изменение масштаба по оси абсцисс;
- Zoom *Y*-asis (масштаб по оси *Y*) – изменение масштаба по оси ординат;
- Autoscale (автоматическое масштабирование) – автоматическая установка оптимального масштаба осей;
- Save current axes settings (сохранение текущих параметров масштаба) – сохранение установленного масштаба осей;
- Properties (свойства) – открытие окна настройки параметров (свойств) блока Scope;
- Print (печать) – печать содержимого окна Scope.

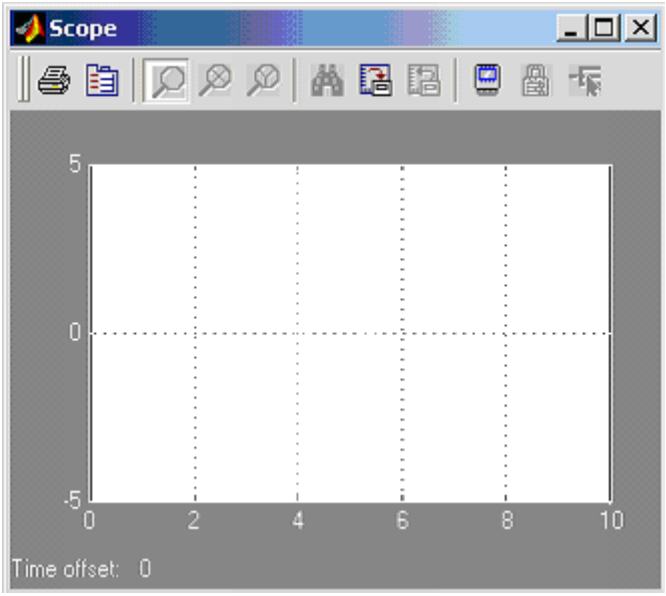


Рис. 2.101. Окно Scope

Окно свойств осциллографа содержит вкладки General и Data History (рис. 2.102). Во вкладке General можно настроить следующие параметры:

- Number of axes – число осей (каналов) осциллографа;
- Time range – пределы временного интервала;
- Tick labels – вывод/скрытие отметок по осям;
- Sampling – установка временных соотношений: Decimation (прореживание в тактах эталонного времени, значение по умолчанию равно 1) или Sample Time (в десятичных долях времени, по умолчанию 0).

Параметр Number of axes позволяет превратить одноканальный осциллограф в многоканальный. При этом осциллограф приобретает несколько входных портов, к которым можно подключить различные сигналы.

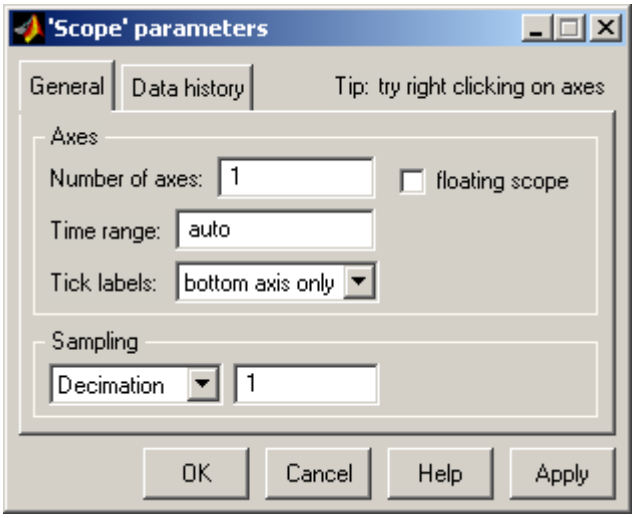


Рис. 2.102. Окно свойств Scope

2.9.1.4. Раздел Sources

Блоки, входящие в раздел Sources (источники), предназначены для формирования сигналов, которые обеспечивают управление работой модели в целом или отдельных ее частей.

Все блоки-источники имеют по одному выходу и не имеют входов. В качестве источников сигналов может использоваться большое количество блоков. Опишем только некоторые из них.

1. *Блок Ramp* (генератор)

- Блок Ramp – генератор линейно возрастающего (убывающего) сигнала вида $F(t) = kt$. Окно настройки параметров блока Ramp представлено на рис. 2.103.

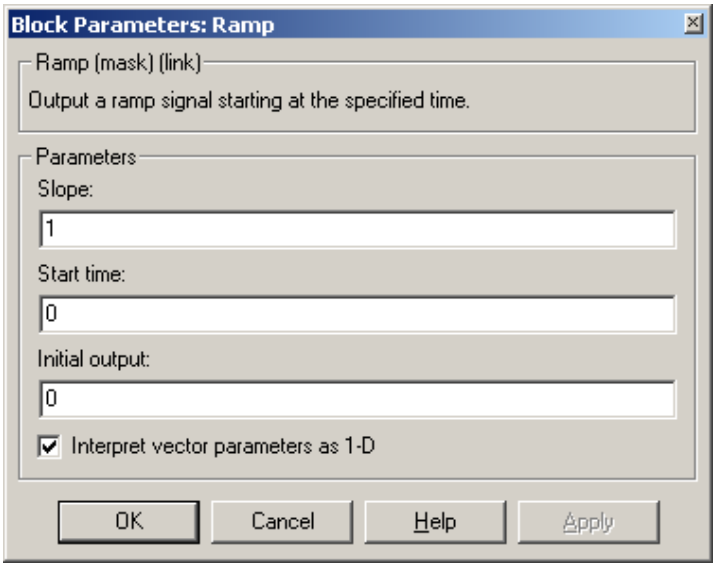


Рис. 2.103. Окно настройки параметров блока Ramp

Параметры источника:

- Slope – угловой коэффициент временной зависимости *k*;
- Start time – время, начиная с которого воздействие возрастает;
- Initial value – начальный уровень воздействия.

2. *Блок Step* («ступенька»)

Блок Step – источник воздействия в виде одиночного перепада; имеет следующие настраиваемые параметры (рис. 2.104):

- Step time – время появления перепада (скачка);
- Initial value – начальное значение воздействия (до перепада);
- Final value – конечное значение воздействия (после перепада);
- Sample time – эталонное время (дискрет по времени). Для непрерывных систем эталонное время равно 0, в то время как для дискретных систем этот параметр определяет шаг дискретизации T_0.

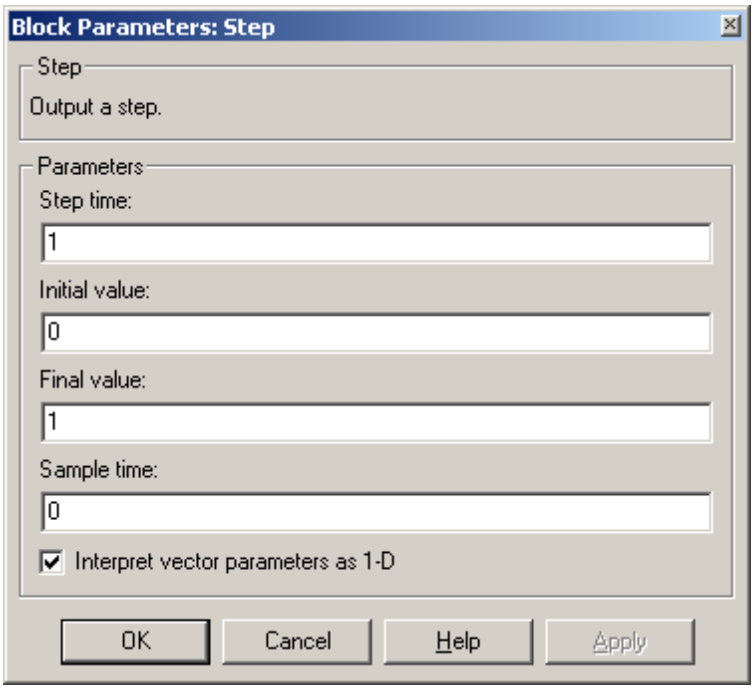


Рис. 2.104. Окно настройки параметров блока Step

3. Блок *Random Number* (случайное число)

Источник случайного сигнала Random Number служит для создания случайного сигнала с равномерным распределением уровня. Окно настройки параметров данного блока представлено на рис. 2.105. Специфическими параметрами этого источника являются среднее значение сигнала Mean и среднеквадратическое отклонение Variance. Назначение двух других параметров уже описывалось выше.

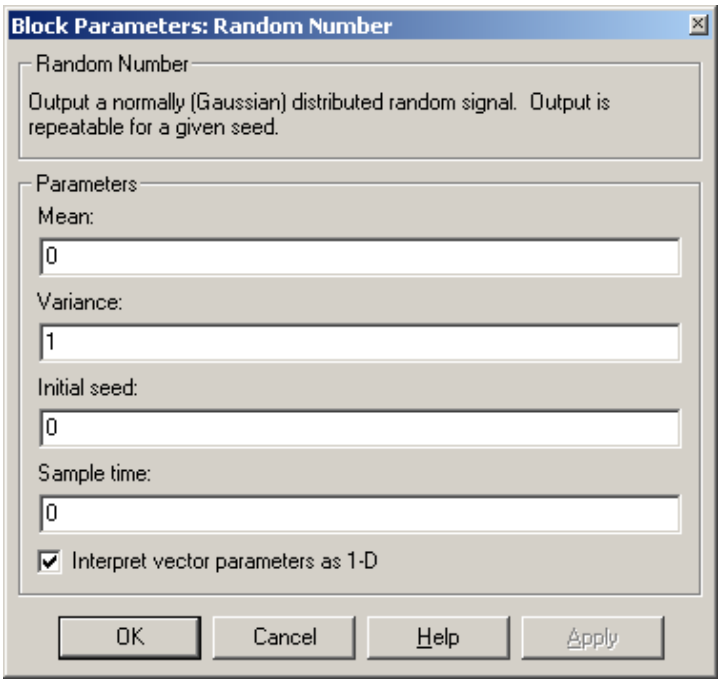


Рис. 2.105. Окно настройки параметров блока Random Number

4. Блок *Uniform Random Number* (равномерное случайное число)

Источник случайного сигнала Uniform Random Number служит для генерации случайного сигнала с равномерным распределением. Уровень сигнала ограничен сверху и снизу значениями Maximum и Minimum. Окно настройки параметров данного блока представлено на рис. 2.106.

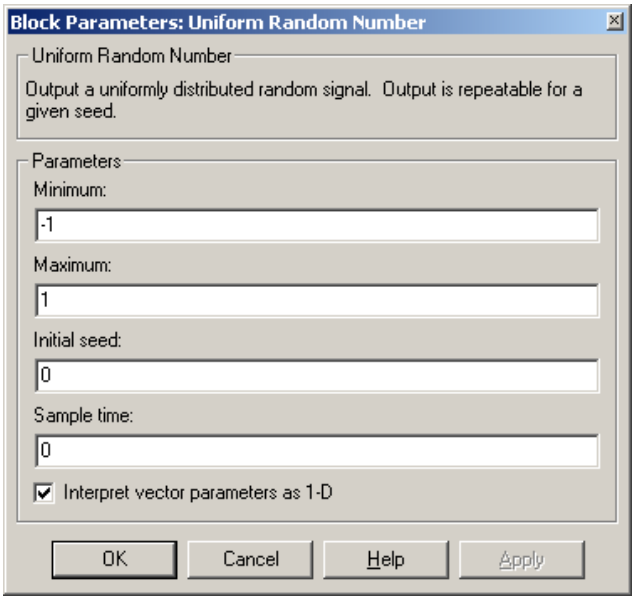


Рис. 2.106. Окно настройки параметров блока Uniform Random Number

2.9.1.5. Библиотека Control System Toolbox

Блоки библиотеки Control System Toolbox представлены на рис. 2.107. С помощью данных блоков можно воспользоваться LTI Viewer – средством для построения различного рода характеристик (переходной, весовой, частотной и т.д.). Для этого необходимо на вход системы поставить блок Input Point, а на

выход – Output Point. Более подробно использование LTI Viewer рассмотрено в главе «Порядок выполнения самостоятельной работы».

Рис. 2.107. Блоки библиотеки Control System Toolbox

2.9.2. Задания для самостоятельной работы

1. В рабочей области пакета MATLAB собрать схему одноконтурной САР, (см. рис. 2.30). Модель ОУ описывается упрощенной формой ПФ ОУ (2.8). Модель регулятора описывается также ПФ динамических звеньев для П-регулятора (2.10), ПИ-регулятора (2.11), ПИД-регулятора (2.61). Числовые значения ПФ ОУ $k_{об}$, $T_{об}$ и $\tau_{об}$ выбрать из табл. 2.13, номер варианта уточнить у преподавателя. Передаточные функции регуляторов собрать как параллельные звенья пропорциональной, интегральной и дифференциальной составляющих. Базовые (оптимальные) настройки регуляторов $k_{р}^{о}$, $T_{И}^{о}$, $T_{Д}^{о}$, определенные методом Циглера-Никольса (см. п. 2.5.2.5), также выбрать из табл. 2.13.

2. Для анализа качества САР во временной области необходимо на вход схемы подать единичную «ступеньку» в виде блока STEP, к выходу подключить блок Scope.

3. С целью определения устойчивости системы провести полный факторный эксперимент (ПФЭ) типа 2^k [2.9]. В качестве факторов эксперимента выбрать $k_{об}$, $T_{об}$ и $\tau_{об}$. Результаты представить в виде семейства переходных характеристик. ПФЭ провести для САР с П-, ПИ-, ПИД-регуляторами отдельно.

4. По переходным характеристикам найти следующие параметры: время регулирования $t_{р}$, максимальное динамическое отклонение σ, перерегулирование η для каждой реализации (см. п. 2.5). Результаты представить в виде обобщенной таблицы. Выявить закономерности влияния варьируемых параметров на показатели качества системы.

5. Результаты моделирования САР представить следующим образом:

а) полученную в рабочей области MATLAB модель одноконтурной САР с конкретными данными ОУ ($k_{об}$, $T_{об}$ и $\tau_{об}$), базовыми настройками регуляторов ($k_{р}^{о}$, $T_{И}^{о}$, $T_{Д}^{о}$), а также базовые выходные параметры переходного процесса (время регулирования, максимальное динамическое отклонение, перерегулирование САР);

б) результаты ПФЭ с моделью системы: семейство переходных характеристик для П-, ПИ-, ПИД-регуляторов отдельно, обобщенную таблицу

результатов (время регулирования, максимальное динамическое отклонение, перерегулирование СУ);

в) выявить и написать выводы о закономерности влияния варьируемых параметров системы ($k_{об}$, $T_{об}$ и $\tau_{об}$) на показатели качества системы (время регулирования, максимальное динамическое отклонение, перерегулирование).

2.9.3. Порядок выполнения работы для схемы с П-регулятором

Собрать схему САР, представленную на рис. 2.108.

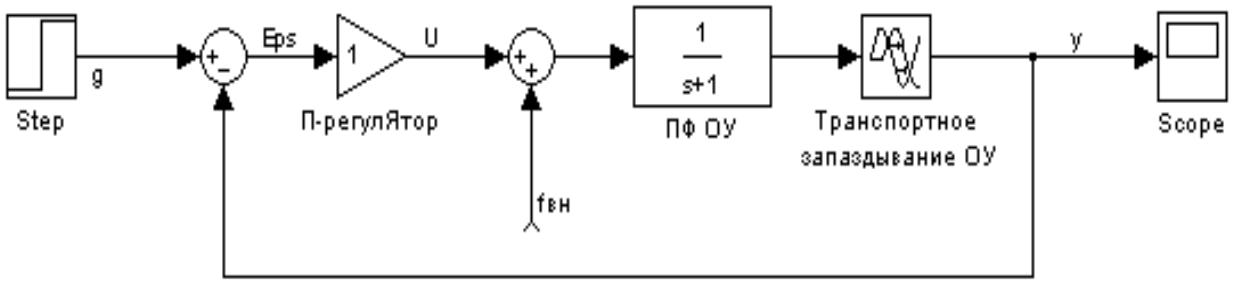


Рис. 2.108. Модель эксперимента с П-регулятором

Для этого необходимо:

1) выбрать блок Gain (коэффициент усиления) в разделе Math, реализующий передаточную функцию П-регулятора;
2) выбрать два сумматора Sum в разделе Math;
3) для реализации объекта управления выбрать два блока:

- блок Transfer Fcn (передаточная функция) в разделе Continuous;
- блок Transport Delay (транспортное запаздывание), реализующий транспортное запаздывание, в разделе Continuous;

4) для анализа качества САР во временной области на вход схемы подать единичный дискретный скачок Step, который можно выбрать в разделе Sources;
5) к выходу схемы подключить Scope (индикатор).

Собирание схемы модели в рабочей области осуществляется путем перетаскивания выбранного из браузера библиотеки Simulink с помощью нажатой левой кнопки мыши.

После составления схемы необходимо задать числовые значения для ПФ П-регулятора и объекта управления. Свойства соответствующих блоков устанавливаются в диалоговом окне Block Parameters (блок параметров), которое вызывается двойным нажатием левой кнопки мыши на соответствующий блок. В окне свойств Gain (рис. 2.109) необходимо указать статический коэффициент усиления П-регулятора.

В окне свойств Transfer Fcn (рис. 2.110) задать Numerator (числитель) ПФ ОУ (запись [1] обозначает, что числитель передаточной функции ОУ имеет значение 1), Denominator (знаменатель) (запись [1 1] означает, что числитель передаточной функции ОУ имеет значение (*s*+1).

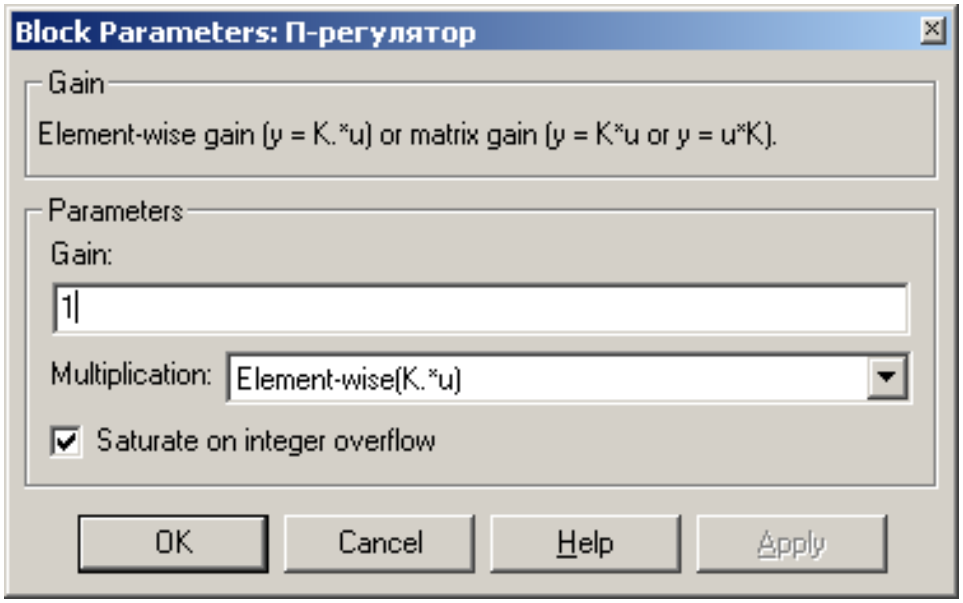


Рис. 2.109. Окно свойств Gain

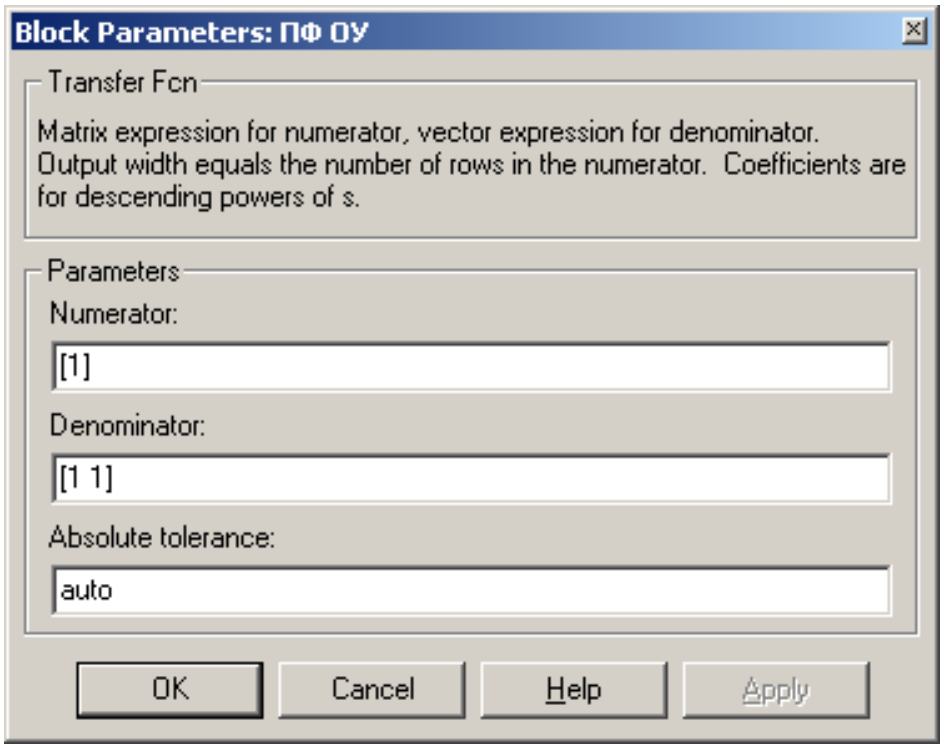


Рис. 2.110. Окно свойств Transfer Fcn

В окне свойств Transport Delay (см. рис. 2.111) в поле Time Delay задается значение величины времени транспортного запаздывания, а также порядок линеаризации Паде.

После задания всех числовых значений необходимо соединить все блоки между собой, как показано на рис. 2.108. Для этого нужно подвести указатель

мыши к выходу предыдущего блока и после появления перекрестия, удерживая левую кнопку мыши, подвести перекрестие к входу последующего блока.

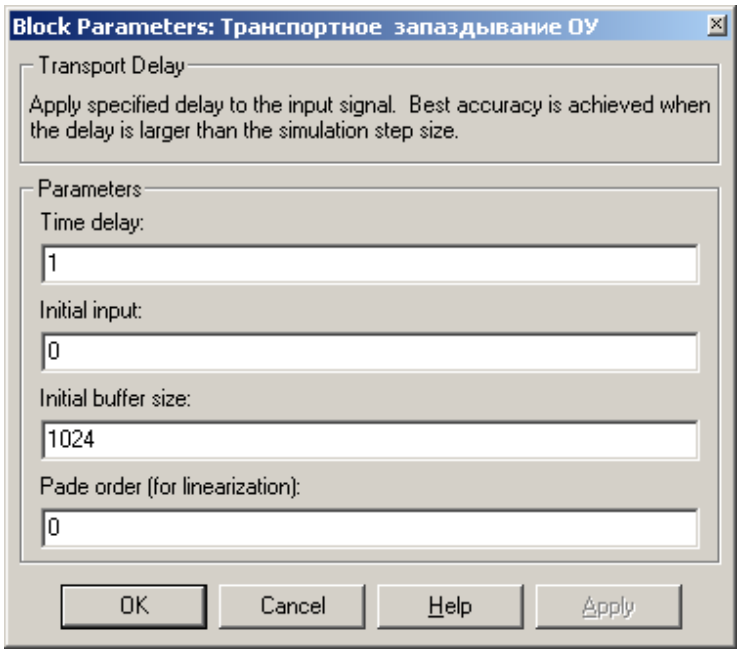


Рис. 2.111. Окно свойств Transport Delay

После соединения всех блоков необходимо замкнуть обратную связь и в окне свойств Sum, представленном на рис. 2.112. В поле List of signs написать следующую последовательность символов: |+-, которая означает, что у блока сумматора имеется два входа – один справа со знаком плюс и один снизу со знаком минус.

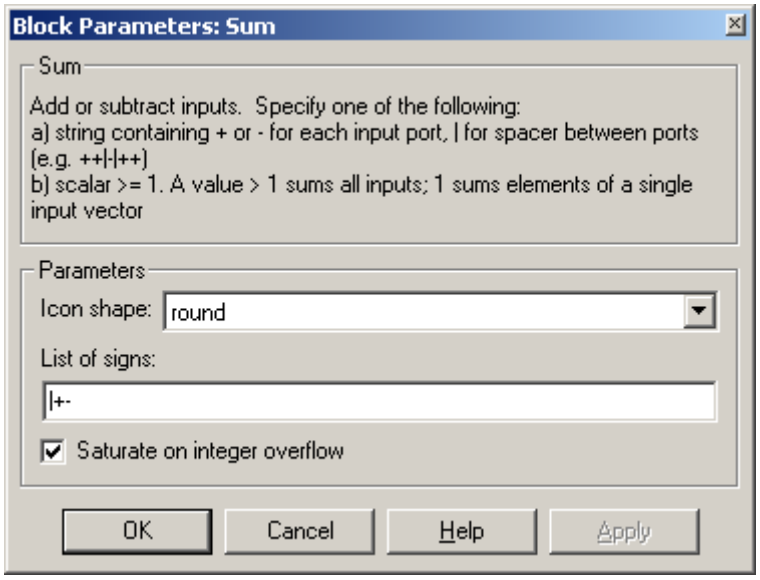


Рис. 2.112. Окно свойств Sum

Анализ САР во временной области можно проводить тремя способами:

1) с помощью индикатора Scoop;
2) с помощью *m*-функций;

3) с помощью специального механизма, который реализует Control System Toolbox в браузере библиотеки Simulink.

Первый способ очень прост в реализации, но при его использовании отсутствует возможность автоматического определения параметров переходного процесса (времени регулирования, перерегулирования и т.д.), а также возможность вставки графиков в отчет без предварительной обработки.

Второй способ также автоматически не позволяет определять параметры переходного процесса, зато дает возможность вставки в отчет графиков в пригодном для распечатывания виде. Однако данный способ требует дополнительных затрат на программирование.

Третий способ снимает все вышеописанные недостатки и является наиболее оптимальным. В браузере библиотеки Simulink Control System Toolbox поочередно выбираются два блока – Input Point (входная точка) и Output Point (выходная точка). Блок Input Point перетаскивается в рабочую область моделирования и вставляется между блоками Step и Sum, блок Output Point – между блоками Transport Delay и Scope, как показано на рис. 2.113.

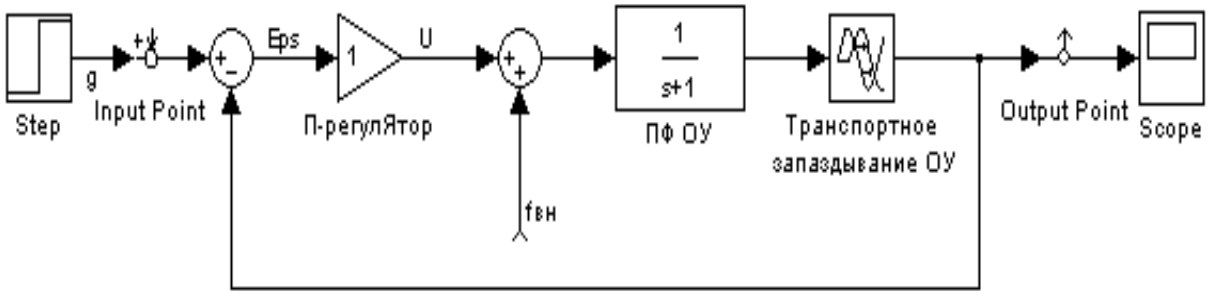


Рис. 2.113. Модель эксперимента с П-регулятором и блоками Control System Toolbox

После этого в окне модели в меню Tools необходимо выбрать пункт Linear Analysis (линейный анализ) и в появившемся окне LTI Viewer в меню Simulink выбрать пункт Get Linearized Model (получить линеаризованную модель). На экране появится окно, представленное на рис. 2.114.

После этого, нажав в области графика правой кнопкой мыши, можно вызвать окно настроек, в котором реализованы следующие функции:

1. Plot Type (тип графика).

Данная опция позволяет вывести на экран следующие основные параметры:

- Step (график переходной функции – реакция на единичный скачок);
- Impulse (график весовой функции – реакция системы на дельта-функцию);
- Bode (построение ЛАЧХ и ЛФЧХ);
- Nyquist (построение амплитудно-фазовой характеристики);
- Pole\Zero (отображение нулей и полюсов ПФ на плоскости).

2. Characteristics (характеристики).

   Данная опция позволяет вывести на экран следующие основные параметры:

- Peak Response (максимальное значение функции реакции на задающее воздействие);
- Setting Time (время регулирования);
- Rise Time (время нарастания);
- Steady State (значение установившегося режима).

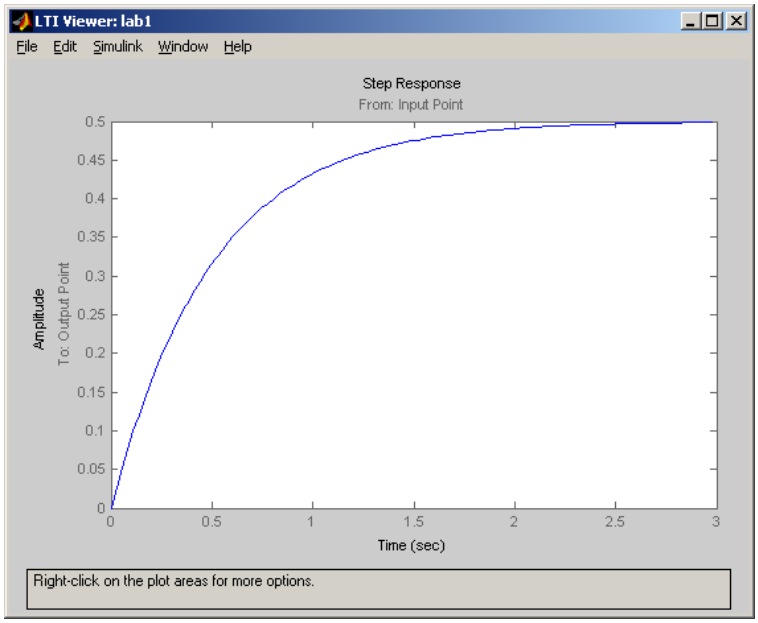


Рис. 2.114. Пример использования LTI Viewer

3. Grid (сетка).

   Позволяет отображать/скрывать сетку в области построения графика.

4. Zoom (изменение масштаба).

   Позволяет изменять масштаб по оси *X* и *Y*.

5. Properties (дополнительные свойства).

   В появившемся окне (см. рис. 2.115) необходимо проделать следующие операции:

- в закладке Labels (метки) задать название графика (Title), название оси *X* (*X*-Label) и название оси *Y* (*Y*-Label);
- в закладке Limits (пределы) имеется возможность задания нижней и верхней границ построения графика по соответствующим осям. Опция Auto-Scale позволяет выполнять автоматическое масштабирование по осям *X* и *Y*;
- в закладке Style (стиль) задаются размер и стили шрифтов надписей на графике;

- в закладке Characteristics (характеристики) задается значение дельта-трубки (по умолчанию 5 %) и значения для определения времени нарастания выходного сигнала (по умолчанию от 10 до 90 %), а также имеется возможность отображения на графике соответствующих значений времени

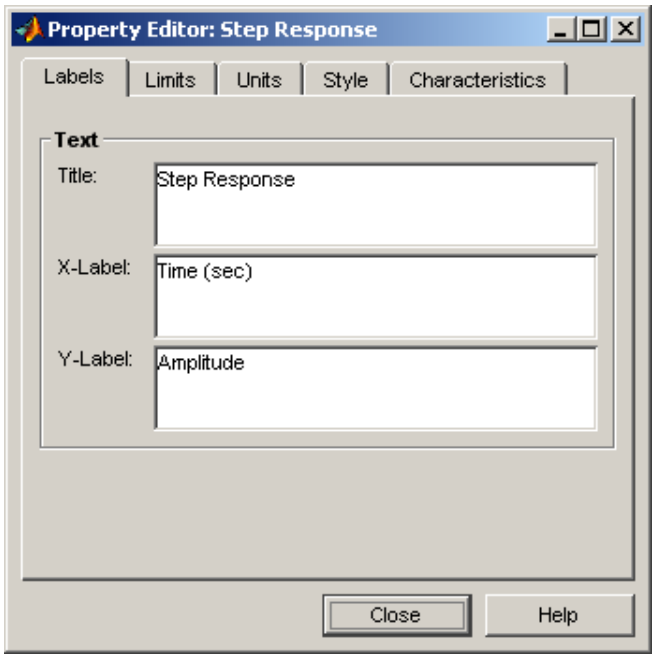


Рис. 2.115. Окно настройки LTI Viewer

регулирования, времени нарастания, максимального значения функции реакции на задающее воздействие (максимального динамического отклонения) и значения установившегося режима.

2.9.4. Пример моделирования одноконтурной САР с П-регулятором

1. Собрать схему САР с П-регулятором, представленную на рис. 2.116, задать числовые значения для ОУ $k_{об}$ = 0,5, $T_{об}$ =24, $\tau_{об}$ = 7,0, для регулятора $k_р^о$ = 6,039. В свойствах блока Transport Delay необходимо задать значение Pade Order, равное 15 (порядок линеаризации), для линеаризации звена транспортного запаздывания.

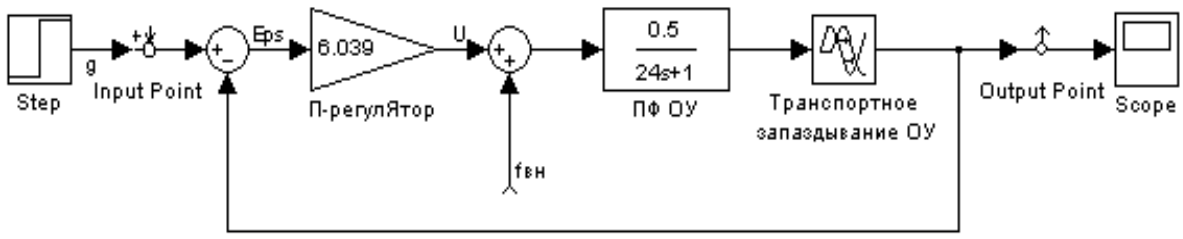


Рис. 2.116. Схема одноконтурной САР с П-регулятором

2. В меню Tools выбрать пункт Linear Analysis.
3. В появившемся окне LTI Viewer в меню Simulink выбрать пункт Get Linearized Model.

4. После этого на экране появится окно, в котором будет отображаться переходная функция при базовых (исходных) значениях, представленная на рис. 2.117.

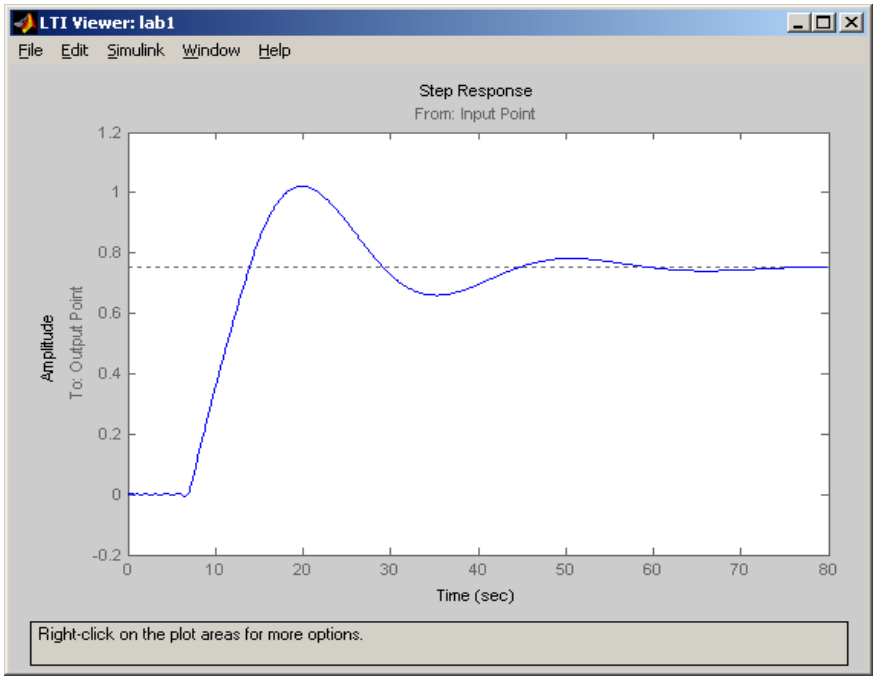


Рис. 2.117. Переходная характеристика при базовых значениях ОУ

5. При каждом следующем эксперименте (изменении числовых значений модели) необходимо повторять пункт 2. Составим план полного факторного эксперимента, представленный в табл. 2.14.

Таблица 2.14

План полного факторного эксперимента

Номер опыта	$k_{об}$	$T_{об}$, с	$\tau_{об}$, с
Базовый	0,5	24	7
1	0,4 (-20 %)	19,2 (-20 %)	5,6 (-20 %)
2	0,4 (-20 %)	19,2 (-20 %)	8,4 (+20 %)
3	0,4 (-20 %)	28,8 (+20 %)	5,6 (-20 %)
4	0,4 (-20 %)	28,8 (+20 %)	8,4 (+20 %)
5	0,6 (+20 %)	19,2 (-20 %)	5,6 (-20 %)
6	0,6 (+20 %)	19,2 (-20 %)	8,4 (+20 %)
7	0,6 (+20 %)	28,8 (+20 %)	5,6 (-20 %)
8	0,6 (+20 %)	28,8 (+20 %)	8,4 (+20 %)

6. После того как проведен полный факторный эксперимент (на экране отображается 8 графиков), необходимо определить максимальное динамическое отклонение и время регулирования с помощью опции

Characteristics. После выбора необходимых параметров на графике переходной характеристики появятся две точки (рис. 2.118). Нажав левой кнопкой мыши на соответствующие точки, можно определить основные показатели качества одноконтурной САР: время регулирования (Setting Time), максимальное динамическое отклонение (Peak amplitude) и перерегулирование (Overshoot).

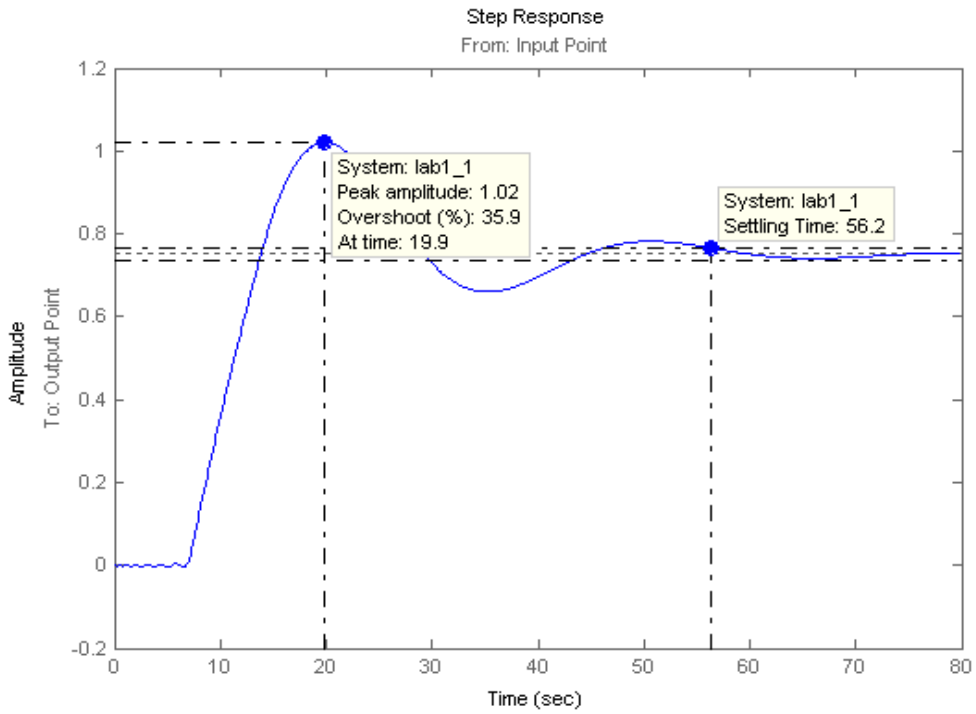


Рис. 2.118. Определение показателей качества

Результаты полного факторного эксперимента при определении качества САР сведем в табл. 2.15.

Таблица 2.15

Результаты полного факторного эксперимента с П-регулятором

Номер опыта	t_p, с	σ	η, %
Базовый	56,2	0,27	35,9
1	35,9	0,16	22,1
2	77,9	0,35	48,0
3	30,0	0,04	4,8
4	53,9	0,16	22,1
5	63,7	0,43	50,2
6	292,0	0,73	88,9
7	37,5	0,21	21,9
8	50,2	0,43	95,5

7. Затем в окне LTI Viewer в меню File выбрать Print to Figure, после чего на экране появится окно, представленное на рис. 2.119.

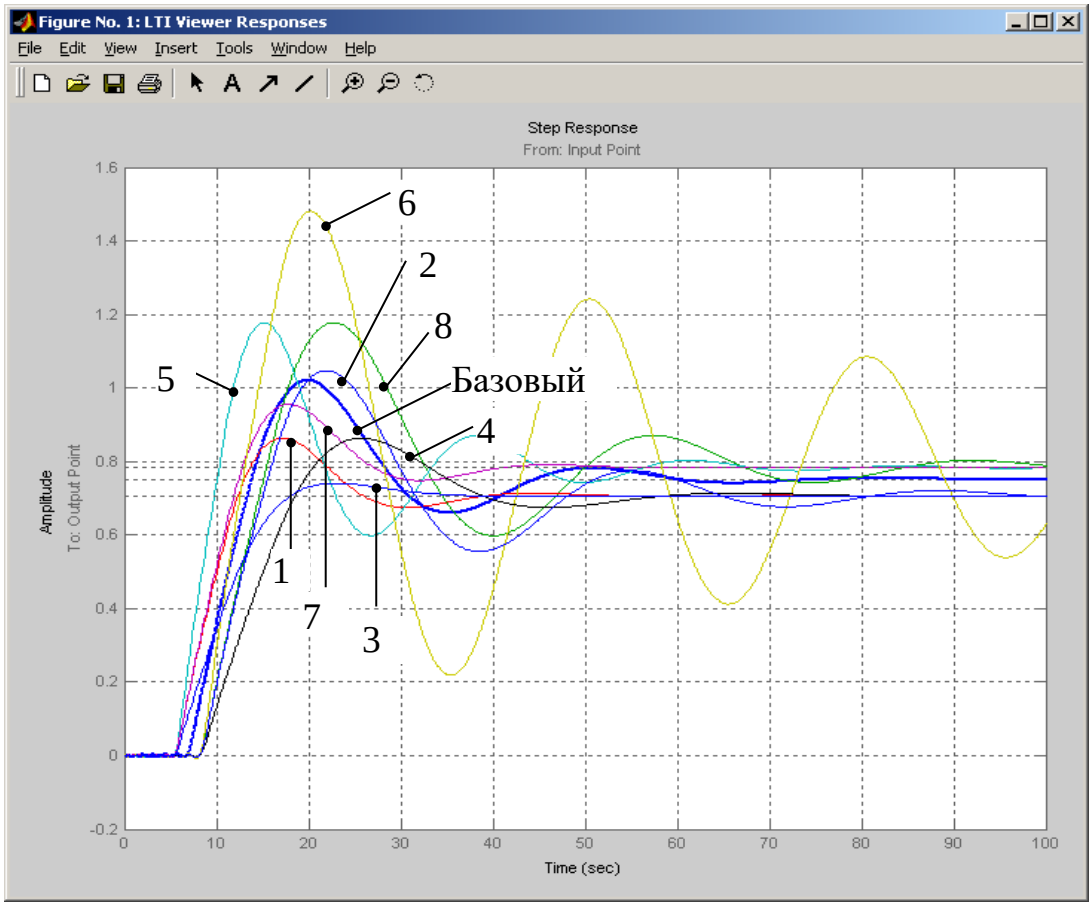


Рис. 2.119. Результаты эксперимента САР с П-регулятором

8. В данном окне в меню Edit выбрать Copy Options, где установить флажок Force white background, затем выбрать Copy Figure, после чего данный график можно представить в отчете.
9. Сделать выводы о закономерности влияния варьируемых параметров системы ($k_{об}$, $T_{об}$ и $\tau_{об}$) на показатели качества системы (время регулирования, максимальное динамическое отклонение, перерегулирование).

2.9.5. Пример моделирования одноконтурной САР с ПИ-регулятором

В рабочей области MATLAB собрать схему с ПИ-регулятором, представленную на рис. 2.120. Изменить вид сумматора можно с помощью окна свойства Sum (см. рис. 2.112), выбрав в Icon shape (форма иконки) вид rectangular (прямоугольная). Задать числовые значения для ОУ $k_{об}$ = 0,5, $T_{об}$ = 24, $\tau_{об}$ = 7,0, для регулятора $k_р^о$ = 5,435, $T_И^о$ = 21,085.

Проведем полный факторный эксперимент в соответствии с табл. 2.14. Результаты полного факторного эксперимента при определении качества САР сведем в табл. 2.16.

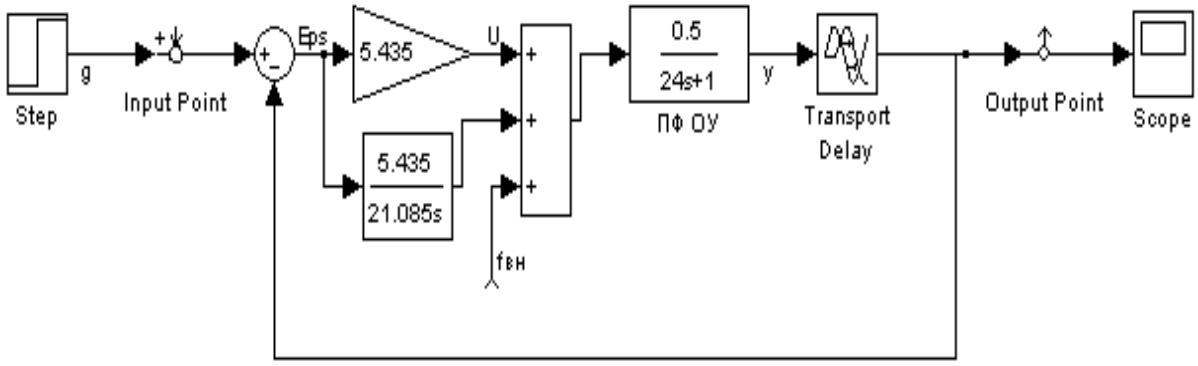


Рис. 2.120. Модель эксперимента с ПИ-регулятором

Таблица 2.16

Результаты полного факторного эксперимента с ПИ-регулятором

Номер опыта	$t_р$, с	σ	η, %
Базовый	66,8	0,35	35
1	43,2	0,11	11
2	90,8	0,40	40
3	63,9	0,10	100
4	64,9	0,28	28
5	61,3	0,42	42
6	401,0	0,87	87
7	36,2	0,24	24
8	116,0	0,59	59

Полученные переходные характеристики одноконтурной САР представлены на рис. 2.121.

2.9.6. Пример моделирования одноконтурной САР с ПИД-регулятором

В рабочей области MATLAB собрать схему с ПИД-регулятором, представленную на рис. 2.122. Задать числовые значения для ОУ $k_{об}$ = 0,5, $T_{об}$ = 24, $\tau_{об}$ = 7,0, для регулятора $k_р^о$ = 7,247, $T_И^о$ = 12,592, $T_Д^о$ = 3,163.

Проведем полный факторный эксперимент в соответствии с табл. 2.14. Результаты полного факторного эксперимента при определении качества САР сведем в табл. 2.17.

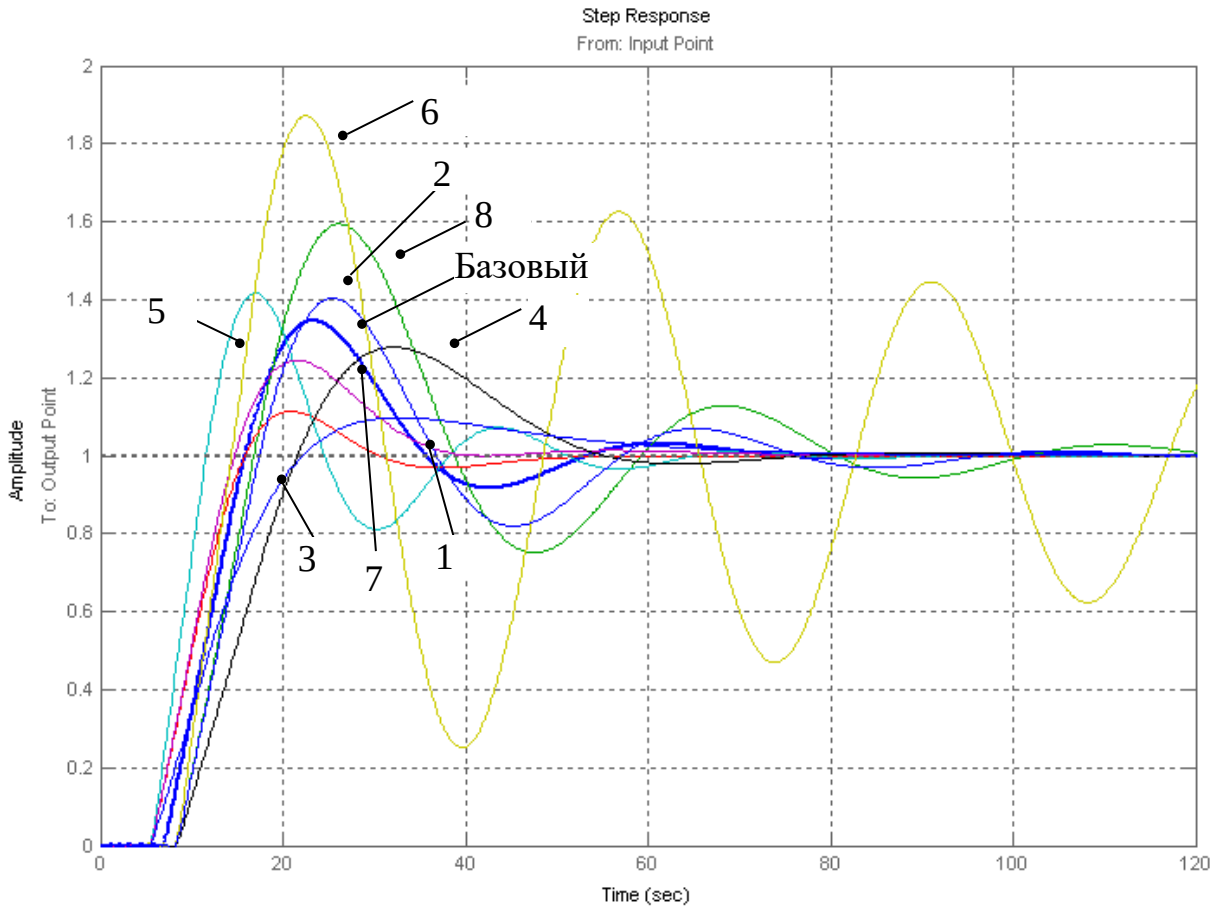


Рис. 2.121. Переходные характеристики одноконтурной САР с ПИ-регулятором

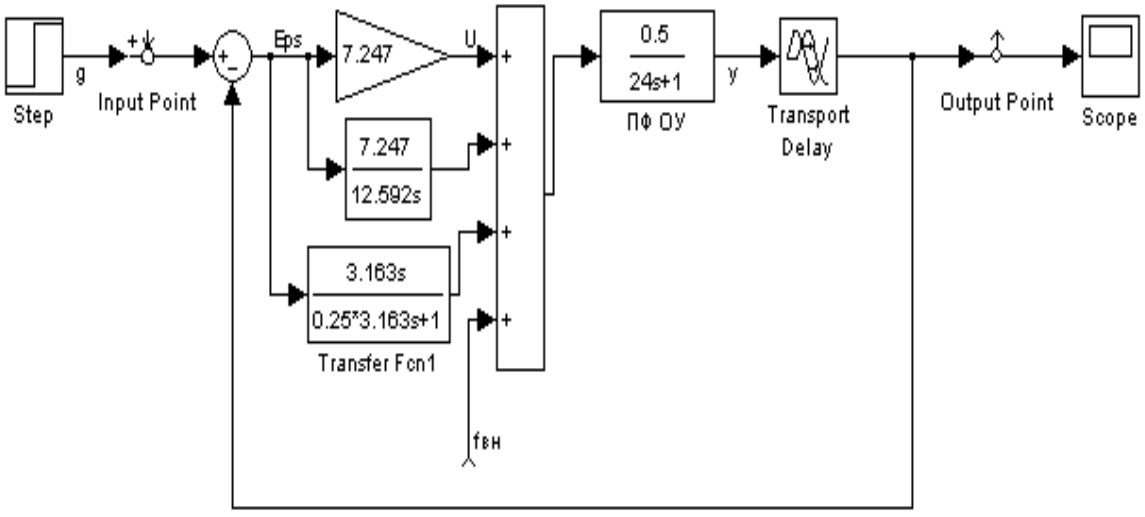


Рис. 2.122. Модель эксперимента с ПИД-регулятором

Полученные переходные характеристики представлены на рис. 2.123.

Как видно из табл. 2.17, в опыте № 6 получился расходящийся процесс. Поэтому приведем переходную характеристику для данного эксперимента отдельно (см. рис. 2.124).

Таблица 2.17

Результаты полного факторного эксперимента с ПИД-регулятором

Номер опыта	$t_р$, с	σ	η, %
Базовый	143	0,84	84
1	56	0,45	45
2	360	1,00	100
3	60	0,37	37
4	156	0,77	77
5	147	0,89	89
6	Расходящийся процесс		
7	58	0,61	61
8	1750	1,24	124

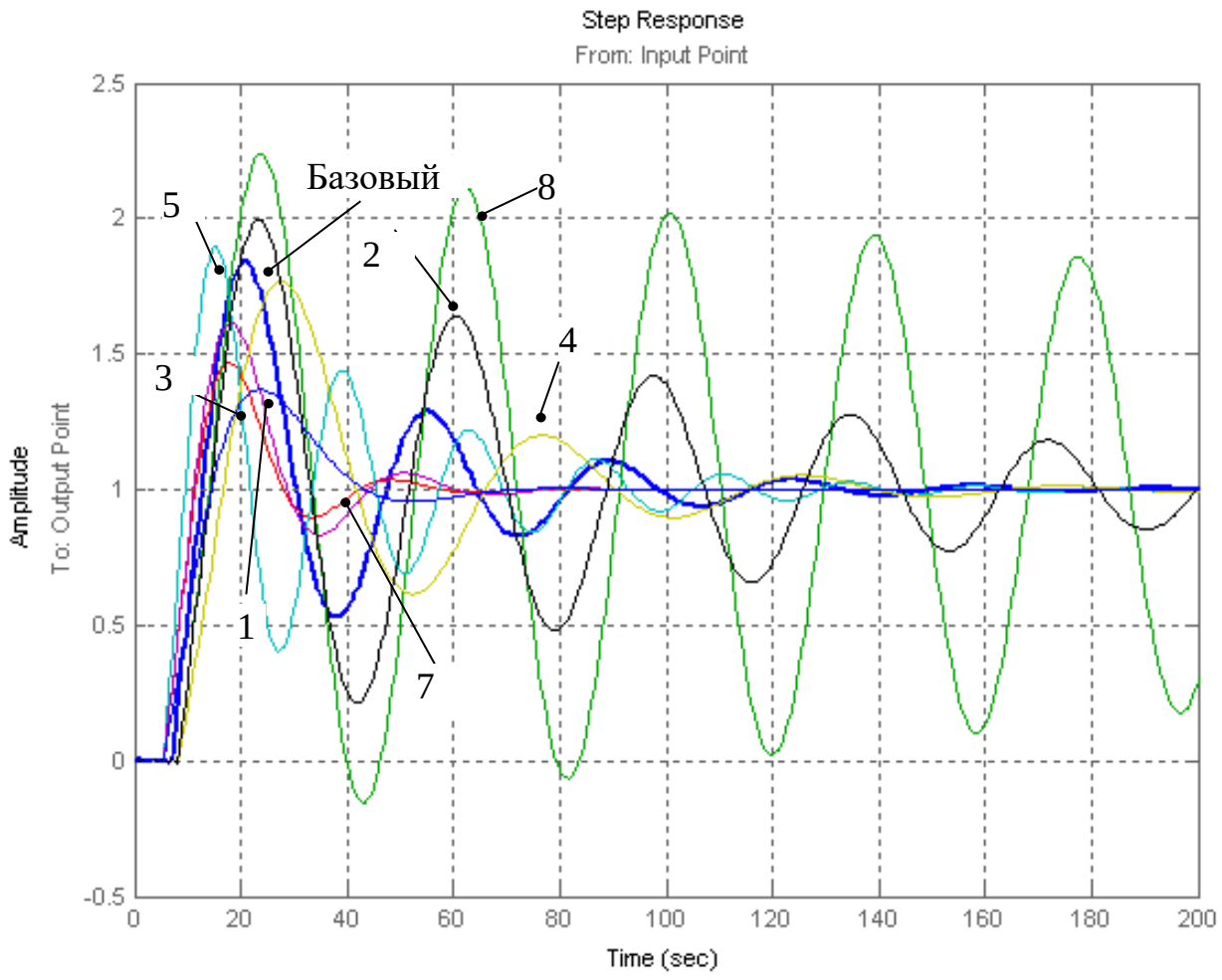


Рис. 2.123. Переходные характеристики одноконтурной САР с ПИД-регулятором

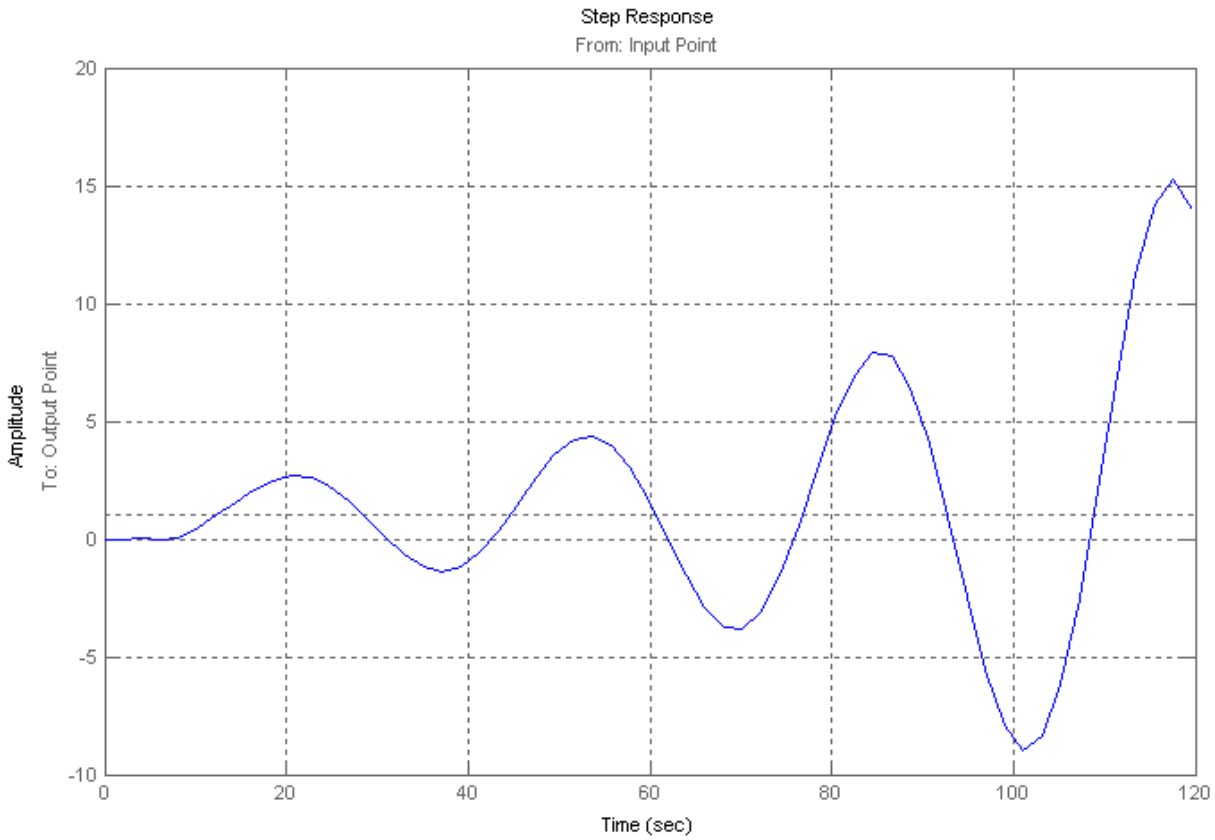


Рис. 2.124. Переходная характеристика одноконтурной САР
с ПИД-регулятором (для опыта № 6)

БИБЛИОГРАФИЧЕСКИЙ СПИСОК К ГЛАВЕ 2

2.1. Томашевский В.Н. Имитационное моделирование в среде GPSS / В.Н. Томашевский, Е. Жданова. М.: Бестселлер, 2003. 412 с.

2.2. Боев В.Д. Моделирование систем: Инструментальные средства GPSS World / В.Д. Боев. СПб. : БХВ-Петербург, 2004. 348 с.

2.3. Шрайбер Т. Дж. Моделирование на GPSS : пер. с англ. / Т. Дж. Шрайбер. М.: Машиностроение, 1980. 592 с.

2.4. Советов Б.Я. Моделирование систем : учеб. пособие для вузов / Б.Я. Советов, С.А. Яковлев. М. : Высш. шк., 2001. 343 с.

2.5. Бесекерский В.А. Теория систем автоматического регулирования / В.А. Бесекерский, Е.П. Попов. 2-е изд., испр. и доп. М.: Наука, 1972. 767 с.

2.6. Бронштейн И.Н. Справочник по математике для инженеров и учащихся втузов / И.Н. Бронштейн, К.А. Семендяев. 15-е изд. М.: Наука, Физматлит, 1998. 608 с.

2.7. Гультяев А. Визуальное моделирование в среде MATLAB: учеб. курс / А. Гультяев. СПб. : Питер, 2000. 432 с.

2.8. Чесноков Ю.Н. Проектирование систем регулирования на ПК: учеб. пособие / Ю.Н. Чесноков, О.А. Гусев. Екатеринбург: УГТУ, 1999. 108 с.

2.9. Дружинина О.Г. Моделирование систем: учеб. пособие / О.Г. Дружинина. Екатеринбург: ГОУ ВПО УГТУ – УПИ, 2005. Ч.2. 196 с.

2.10. Владимировский Б.М. Математика. Общий курс / Б.М. Владимировский, А.Б. Гопрстко, Я.М. Ерусалимский. СПб.: Лань, 2002. 960 с.

2.11. Шеннон Р. Имитационное моделирование систем – искусство и наука: пер. с англ. / Р.Шеннон. М.: Мир, 1978. 418 с.

3. АНАЛИТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ТРАНСПОРТНОЙ СИСТЕМЫ МЕГАПОЛИСА

В данной главе рассматриваются проблемы моделирования системы общественного транспорта.

3.1. АКТУАЛЬНОСТЬ МОДЕЛИРОВАНИЯ

Современное общество нуждается в постоянном увеличении объема транспортного сообщения в целом и движения общественного транспорта в частности, повышаются требования к надежности, безопасности и качеству транспортных потоков. Это требует увеличения затрат на улучшение инфраструктуры транспортной сети, превращения ее в гибкую, хорошо управляемую логистическую систему. При этом риск инвестиций значительно возрастает, если не учитывать закономерности развития транспортной сети, распределение загрузки ее участков. Игнорирование этих закономерностей приводит к частому образованию транспортных пробок, перегрузке/недогрузке отдельных линий, узлов сети и транспортных единиц, повышению уровня аварийности, экологическому ущербу [3.1].

Совершенствование управления системой городского пассажирского транспорта (ГПТ) должно сопровождаться отслеживанием изменений во всех ее компонентах. Необоснованное применение достаточно сложных экспериментов над реальными системами ГПТ может иметь отрицательные последствия. Развитие ГПТ признано в мире первоочередной и наиболее эффективной мерой борьбы с автомобильными заторами. «Легковой автомобиль по использованию площадей транспортных магистралей имеет наихудшие показатели» [3.2]. Провозная способность полосы ГПТ (в зависимости от вида транспорта и интенсивности движения) в 10-100 раз выше, чем провозная способность полосы личного автотранспорта

Поэтому необходимо моделирование систем ГПТ. По определению Шеннона [3.3]: «Модель – это представление объекта, системы или идеи в некоторой форме, отличной от самой целостности». Главной характеристикой модели можно считать упрощение реальной жизненной ситуации, к которой она применяется.

При рациональном решении проблем желательно выявить наиболее существенные и значимые явления и факторы.

С этой целью необходимо формировать модели ГПТ, основанные на абстрагировании от несущественных факторов и упрощающих реальную систему с сохранением ее основных свойств.

Теория транспортных потоков развивалась исследователями различных областей знаний – физиков, математиков, специалистов по исследованию операций, транспортников, экономистов. Накоплен большой опыт

исследования процессов движения. Однако практическое использование теоретических исследований ограничено в силу следующих факторов:

1) транспортный поток нестабилен, получение объективной информации о нем является наиболее сложным и ресурсоемким элементом системы управления;
2) критерии качества управления дорожным движением противоречивы: необходимо обеспечивать бесперебойность движения, одновременно снижая ущерб от движения, накладывая ограничения на скорость и направления движения;
3) дорожные условия, при всей стабильности, имеют непредсказуемые как в части отклонения погодно-климатических параметров так и, собственно, дороги;
4) исполнение решений по управлению дорожным движением всегда неточно при реализации и, учитывая природу процесса дорожного движения, приводит к непредвиденным эффектам [3.4].

Таким образом, трудности формализации процесса движения транспортного потока, в том числе движения общественного транспорта, стали серьезной причиной отставания результатов научных исследований от требований практики.

В этих условиях невозможно обойтись без математических моделей и численных экспериментов, ограничившись лишь результатами инженерных расчетов и экспертными оценками. Моделирование системы общественного транспорта обладает следующими особенностями:

1) при увеличении спроса на услуги перевозки возможно перераспределение транспорта с изменением графиков;
2) перевозки напрямую связаны с непредсказуемостью поведения отдельного пассажира в выборе маршрута, и прочее;
3) большое влияние случайных факторов (ДТП, погода и прочее) и флуктуаций, связанных с сезонами, выходными и праздничными днями и т.п.

3.2. ИСТОРИЯ РАЗВИТИЯ МОДЕЛИРОВАНИЯ ТРАНСПОРТНЫХ ПОТОКОВ

Основы математического моделирования закономерностей дорожного движения были заложены в 1912 году русским ученым, профессором Г.Д. Дубелиром.

Первостепенной задачей, послужившей развитию моделирования транспортных потоков, стал анализ пропускной способности магистралей и перекрестков. В настоящее время пропускная способность является важнейшим критерием оценки качества функционирования путей сообщения.

Первая макроскопическая модель, в которой движение транспортного потока рассматривалось с позиций механики сплошной среды, была предложена в 1955 году Лайтхиллом (*Ligthill*) и Уиземом (*Whitham*) [3.5, 3.6]. Они показали, что методы описания процессов переноса в сплошных средах могут быть использованы для моделирования заторов.

Выделение математических исследований транспортных потоков в самостоятельный раздел прикладной математики впервые было осуществлено Ф. Хейтом [3.7].

В 60 – 70-е годы вновь возник интерес к исследованию транспортных систем. Эта заинтересованность проявилась, в том числе, в финансировании многочисленных контрактов, обращении к авторитетным ученым – специалистам в области математики, физики, процессов управления, таким как Нобелевский лауреат И. Пригожин, специалист по автоматическому управлению М. Атанс, автор фундаментальных работ по статистике Л. Брейман. В нашей стране движение транспорта активно изучалось в конце 70-х годов в связи с подготовкой к Олимпийским играм 1980 года в Москве. Результаты этих исследований неоднократно докладывались на научно-исследовательском семинаре И.И. Зверева на механико-математическом факультете МГУ им. М.В. Ломоносова [3.8].

В конце 80-х начале 90-х годов в США проблемы исследования транспортных систем были возведены в ранг проблем национальной безопасности. К решению этой задачи были привлечены лучшие «физические умы» и компьютерная техника Национальной исследовательской лаборатории Лос-Аламос – *Los Alamos National Lab* (*LANL*).

Сегодня имеется обширная литература по изучению и моделированию транспортных потоков. Несколько академических журналов посвящены исключительно динамике транспортного движения. Наиболее крупными являются *Transportation Research, Transportation Science, Mathematical Computer Simulation, Operation Research, Automatica, Physical Review E, Physical Reports.* Количество публикуемых статей исчисляется сотнями.

3.3. НАУЧНЫЕ ПОДХОДЫ К ИЗУЧЕНИЮ ТРАНСПОРТНЫХ ПОТОКОВ

Математические модели применяются для решения очень широкого круга задач, связанных с транспортом. Рассмотрим сначала модели, предназначенные для прогноза транспортных и пассажирских потоков в сетях с известной геометрией и характеристиками и при известном размещении различных объектов на территории города или городской агломерации. Модели этого типа применяются для поддержки решений в области планирования развития города, для анализа последствий тех или иных мер по организации движения, выборе альтернативных проектов развития транспортной сети и др.

Для прогноза транспортных потоков необходимо ответить на вопрос: сколько и по каким путям будет совершаться передвижений в транспортной сети в то или иное время суток. Ответив на этот вопрос, можно определить

загрузку любого элемента сети. Общее количество передвижений из каждого района города в каждый другой район принято называть межрайонной корреспонденцией. Задача моделирования транспортных потоков традиционно разделяется на две взаимосвязанных задачи. Во-первых, необходимо оценить межрайонные корреспонденции. Во-вторых – распределить корреспонденции по транспортной сети. Это означает, что для каждой пары районов нужно предсказать, по каким путям будут осуществляться передвижения, и сколько участников движения будет использовать тот или иной путь. Охарактеризуем основные идеи решения этих задач.

3.3.1. Моделирование корреспонденций

Исторически одной из первых математических моделей, предложенных для оценки межрайонных корреспонденций, была гравитационная модель. Она основана на следующем простом положении: корреспонденция из одного района в другой будет тем больше, чем больше емкости районов прибытия и отправления, и чем ближе друг к другу расположены эти районы. Название модели объясняется схожестью формулировки с законом всемирного тяготения, согласно которому сила притяжения пропорциональна массам тел и обратно пропорциональна квадрату расстояния между ними. Роль масс здесь играют емкости районов (можно понимать их как общие объемы прибытия и отправления в этих районах). Близость или дальность районов определяется не географическим расстоянием, а дальностью в «транспортном» смысле. Точное определение транспортной дальности может отличаться в разных моделях. С интуитивной точки зрения эта величина должна отражать степень удаленности районов с учетом времени и удобства передвижений, предоставляемых транспортной сетью.

Хотя межрайонные корреспонденции сами по себе также представляют интерес для планировщика, однако основная цель их вычисления – расчет транспортных потоков путем распределения корреспонденций по конкретным путям в транспортной сети. Далее рассмотрены основные подходы к решению этой задачи.

3.3.2. Модель равновесного распределения потоков

Основным предположением в задаче распределения корреспонденций по сети является то, что каждый участник движения стремится добраться до цели как можно быстрее (в более общей формулировке – стремится минимизировать обобщенную цену пути). Это на первый взгляд тривиальное предположение приводит к трудностям при его реализации. Действительно, для выбора кратчайшего пути необходимо знать время движения на всех элементах сети. Однако время движения на каждом элементе сети зависит от загрузки этого элемента, в частности, оно увеличивается при возрастании загрузки. Таким образом, в задаче возникает обратная связь: для правильного распределения корреспонденций по путям нужно знать загрузку сети, а сама эта загрузка возникает как результат распределения.

Анализ этой ситуации приводит к концепции равновесного распределения потоков. Согласно этой концепции в транспортной системе постоянно происходит «игра», в ходе которой водители пробуют те или иные пути движения, исходя из предшествующего опыта. Выбирая те или иные пути, водители косвенно влияют друг на друга через формирование загрузки сети. В результате этого процесса в системе устанавливается некоторое равновесное состояние, суть которого может быть кратко выражена в следующем предложении: при равновесном распределении ни один из участников движения не может изменить свой путь так, чтобы уменьшить свое индивидуальное время поездки, и, следовательно, не имеет мотивации к изменению своего пути.

Исследования показывают, что равновесное распределение обладает нетривиальными свойствами, которые заслуживают обсуждения. Заметим, прежде всего, что равновесное распределение по своему определению не является оптимальным в смысле какого-либо общесистемного критерия. Оно формируются как результат оптимизации водителями своих индивидуальных критериев и поэтому называется «оптимальным для пользователей». В противоположность этому можно рассчитать «системно оптимальное» распределение, то есть такое, при котором общие суммарные затраты времени были бы минимальными. Тот факт, что равновесное распределение не всегда является системно оптимальным, понятен. Действительно, возможны ситуации, когда общие затраты времени могли бы снизиться, если бы часть водителей согласилась ехать по не самым выгодным для них путям, облегчая движение для других (большего числа!) водителей. Удивительным является тот факт, что в некоторых случаях все водители при равновесном распределении добираются до мест назначения дольше, чем это было бы при системно оптимальном распределении. Ситуация выглядит так: пусть в начальный момент времени некий руководящий орган предписал всем водителям пути движения, при которых обеспечивается системный оптимум, а затем предоставил им возможность менять пути по своему усмотрению. Некоторые водители сразу же обнаружат, что в сложившихся условиях они могли бы выиграть во времени, избрав другие пути. Изменение путей приведет к изменению «сложившихся условий» (то есть загрузки сети) и, следовательно, к дальнейшему изменению путей. В конечном итоге этот дрейф распределения придет к равновесию. При этом может оказаться, что в результате проиграла не только система в целом, но и каждый водитель в отдельности.

3.3.3. Моделирование пассажирских потоков

Равновесное распределение в основном применяется для моделирования автомобильных потоков. Процесс формирования пассажирских потоков в системе общественного транспорта имеет свои особенности. В частности, пользователь общественного транспорта часто принимает решение не о конкретном пути до цели, а скорее о стратегии своего поведения при достижении цели. Например: «Я прихожу на остановку и жду один из

автобусов № 23 или № 28. Если первым придет № 23, еду до станции метро *A*. Иначе – до станции метро *B*. Путь, по которому фактически осуществляется передвижение, при этом не детерминирован, хотя моделируются вполне «устоявшиеся» регулярные передвижения. Понятие кратчайшего пути в этой ситуации заменяется понятием «оптимальной стратегии», то есть стратегии, обеспечивающей, в среднем, наискорейшее достижение цели. Соответственно, основанная на этом понятии модель распределения пассажирских потоков называется моделью оптимальных стратегий.

Не только время передвижения влияет на выбор того или иного пути. Но и, например, денежные затраты, связанные с передвижением, а также другие факторы, выражающие удобство и привлекательность того или иного передвижения. Часто эти факторы можно учесть, переводя их в условные минуты, и добавляя к времени движения. Сумму настоящего и условного времени называют «обобщенной ценой пути». Все описанные модели сохраняют силу, если вместо времени использовать обобщенную цену.

3.4. ТЕНДЕНЦИИ РАЗВИТИЯ ГОРОДСКОЙ ДОРОЖНОЙ СЕТИ

В начале 2007 года в статье, опубликованной в журнале Physical Review Letters, физики Марк Бартелеми (*Marc Barthelemy*) и Алессандро Фламмини (*Alessandro Flammini*) сообщили, что создана несложная модель, хорошо описывающая развитие городской транспортной сети.

Исследователи отмечают, в последнее время между сетями улиц различных городов найдены удивительные статистические сходства. Это позволяет предположить, что механизм роста дорожной сети в целом описывается общим механизмом, не зависящим от деталей. Именно его и моделируют Бартелеми и Фламмини.

В качестве модели дорожной сети они используют плоский граф, ребра которого представляют улицы, а вершины – их пересечения. Модель состоит из растущей сети дорог и постоянно появляющихся «центров» – новых домов, магазинов и так далее – которые нуждаются в подсоединении к сети. Новые участки дорог (фиксированной длины) появляются через промежутки времени r, центры – через заданные промежутки времени c, $c>r$.

Основным механизмом, управляющим ростом сети, является «принцип локальной оптимальности», который формулируется примерно следующим образом. Если два центра A и B должны быть присоединены к сети и ближайшие к ним точки сети (соответственно M_1 и M_2) не совпадают, то строятся два отрезка дороги M_1A и M_2B. Если же они совпадают в точке M, то сначала строится отрезок MM', такой чтобы сумма расстояний от A и B до M' была как можно меньше суммы расстояний от A и B до M.

В начале система состоит из нескольких центров, соединенных дорогами. Затем через каждый промежуток c к ней добавляется n новых центров (их расположение определяется случайным образом), и дороги начинают расти в

соответствии с описанным алгоритмом (рис. 3.1).

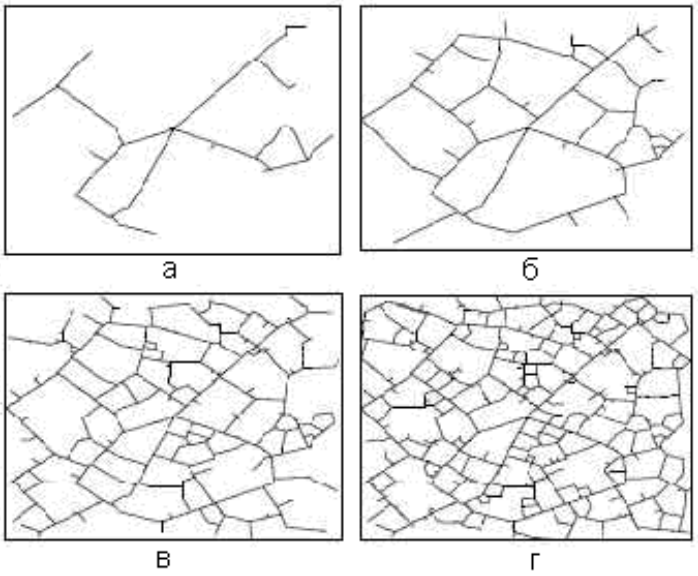


Рис. 3.1. Городская дорожная сеть в различные моменты времени *t*:
а) *t* = 100; б) *t* = 500; в) *t* = 2000; г) *t* = 4000; *n*/*c* = 0,1

Разумеется, сеть улиц каждого города зависит от его географических, исторических и социально-экономических особенностей, поэтому модель не воспроизводит никаких конкретных сетей. Она, по мнению исследователей, хорошо воспроизводит статистические макропараметры, свойственные многим реальным городским сетям (например, отношение общей длины дорог к числу перекрестков).

3.5. СУЩЕСТВУЮЩИЕ ТЕХНИЧЕСКИЕ РЕШЕНИЯ МОДЕЛИРОВАНИЯ ТРАНСПОРТНЫХ ПОТОКОВ

3.5.1. Семейство систем PTV Vision

Программных решений для моделирования транспортных потоков в масштабах мегаполиса и выше не так много. Одним из известных продуктов такого рода является немецкое семейство систем планирования в транспортной инфраструктуре *PTV Vision* (*www.ptv-vision.com, www.ptvamerica.com*), которые используют модели *VISSIM/VISUM*, разработанные почти тридцать лет назад. В состав линейки входит четыре приложения.

Первое – *PTV Simulation*, пожалуй, наиболее интересно, так как используется для микро- и макроскопического моделирования потоков личного, общественного и грузового транспорта, пешеходного движения, настройки работы светофоров в зависимости от заданных параметров, анализа заторов и трехмерных динамических визуализаций перекрестков и развязок. В общем, система предназначена для создания и проигрывания комплексных сценариев развития транспортной системы в зависимости от изменения отдельных ее составляющих.

Разработчики называют среди пользователей системы проектировщиков, чиновников транспортных министерств и ведомств, инженерные компании и т. д. Программное обеспечение применяют для подготовки обоснования постройки или реконструкции дороги, выбора наиболее приемлемого варианта с учетом инженерных и финансовых ограничений и генерации сопутствующих документов для инвестора. По данным компании, ее софт используют более двух тысяч организаций в 75 странах мира. Россия – не исключение. *PTV Vision* применялся при проектировании Третьего транспортного кольца в Москве, Западного скоростного диаметра и перехватывающих парковок в Петербурге, а также проектов в Томске и Иркутске.

Итак, в чем же заключается работа с моделями *PTV*? Моделирование *VISSIM* предполагает микроскопический расчет движения транспорта и проверку работы сигнальных устройств для выбора оптимальной организации движения на перекрестке и оценки пропускной способности для каждого варианта движения (рис. 3.2). При этом учитывается движение в зоне остановок с учетом приоритета общественного транспорта.

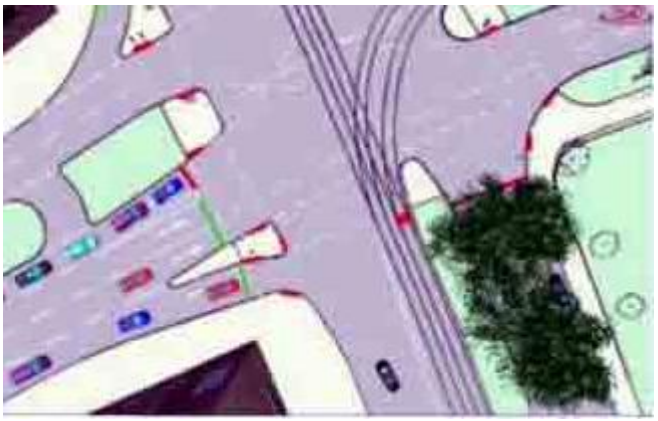


Рис. 3.2. Графическое представление перекрестка в модели VISSIM

На этапе *VISSIM*-моделирования производится анализ «узких» мест. В качестве исходных данных на микроуровне берется растровая основа (карты города, аэрофотосъемки и пр.) и информация о существующей структуре движения. Анализ может производиться по таким параметрам, как нагрузка на дорогу, средняя скорость потока, время поездки и задержек в пути, длина пробок и количество остановок.

VISUM – это макромоделирование существующих и прогнозируемых транспортных потоков с анализом и оценкой правил и интенсивности движения и отработкой сценариев типа «что будет, если…». Здесь в качестве исходных данных принимается сеть путей движения различных видов транспорта (см. рис. 3.3), свойства сети, правила движения и так называемая матрица транспортных передвижений, состоящая из данных «транспортного предложения» и «транспортного спроса».

В результате на первом этапе моделирования формируются две независимых модели. Первая модель транспортного спроса содержит такие данные, как цели и число поездок и кривая транспортного спроса. Вторая

модель сети основана на информации о транспортных системах, ячейках, узлах и остановках.

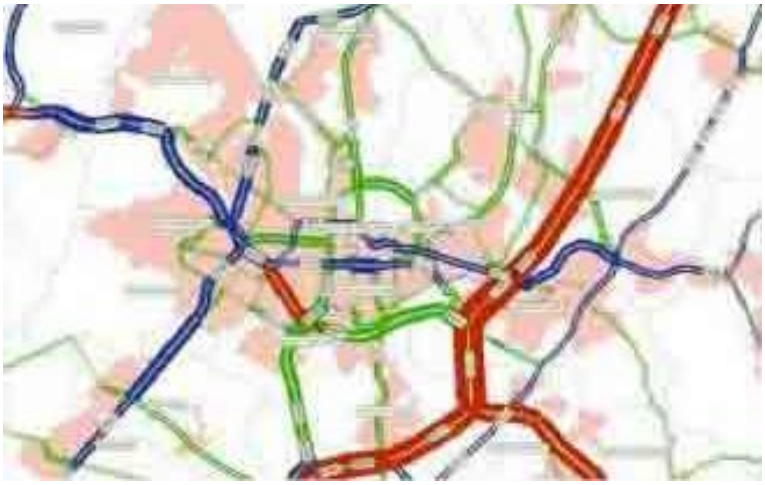


Рис. 3.3. Графическая обработка сети в модели VISUM

Программа обрабатывает эти две модели и на выход выдает модель взаимодействия, которая содержит данные о распределении транспортных потоков автомобилей, о создании маршрутов общественного транспорта и расчета их эффективности, а также по экологическим параметрам (выброс вредных веществ, шумовые воздействия).

3.5.2. Использование PTV Vision в России

Приведем пример использования описанной выше системы в нашей стране.

В 2006 году ООО «Пермгражданпроект» совместно с Пермским техническим университетом проводил работы по компьютерному моделированию дорожного движения на реконструируемых участках улиц Героев Хасана и шоссе Космонавтов города Перми с целью прогнозирования проблем управления транспортными потоками.

Метод компьютерного моделирования применялся в Перми впервые. Он заключается в создании подробнейшей модели улично-дорожной сети (УДС) и имитации всех ситуаций, происходящих на дорогах.

В качестве среды моделирования был использован программный продукт немецкой компании *PTV AG – PTV Vision VISSIM*. Данная система, как было сказано выше, позволяет осуществлять весь спектр задач планирования транспортных систем и организации дорожного движения.

Программа предоставляет средства анализа данных и изучения сценариев, инструментарий для нахождения решений, отвечающих любым требованиям оптимальности.

Первым шагом моделирования является создание геометрической модели. На создание геометрической модели нужно обратить наибольшее внимание, потому что от модели требуется полное сходство с реальной улично-дорожной

сетью. Для начала выбирается топографическая основа, на которой будет размещаться вся УДС. Для получения точной модели сети в *VISSIM* нужно использовать как минимум одну географическую карту с расширением **.bmp*. На рис. 3.4 приведены образцы, которые могут служить топоосновой в *VISSIM*.

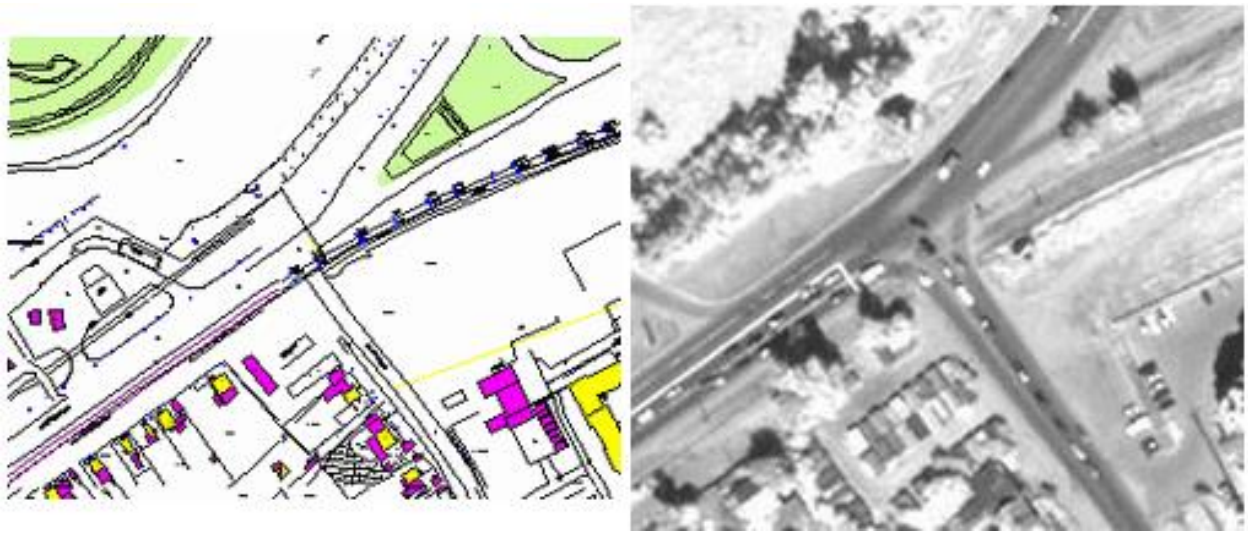


Рис. 3.4. Образцы основы для моделирования перекрестка
шоссе Космонавтов – ул. Свиязева

После того как топооснова выбрана, указывается реальный масштаб. Чтобы масштаб был точный, желательно на топооснове иметь какие-нибудь объекты, реальные размеры которых известны.

Второй этап – построение улично–дорожной сети. Построение УДС состоит из следующих частей:

- нанесение существующей или проектной дорожной сети;
- нанесение остановочных пунктов;
- нанесение парковочных площадей.

Главными элементами дорожной сети являются соединительные отрезки. Порядок нанесения отрезков таков:

- задать главные магистрали;
- установить число направлений на перекрестке;
- установить количество полос на отрезке и на перекрёстке соответственно;
- установить число полос на поворотах.

После того как геометрическая модель готова, следующими этапами моделирования являются задание параметров транспортных потоков, расстановка приоритетов движения, моделирование работы светофоров и движения общественного транспорта. Когда все эти пункты выполнены, можно считать модель готовой и переходить к её непосредственному анализу. Готовая модель представлена на рис. 3.5.

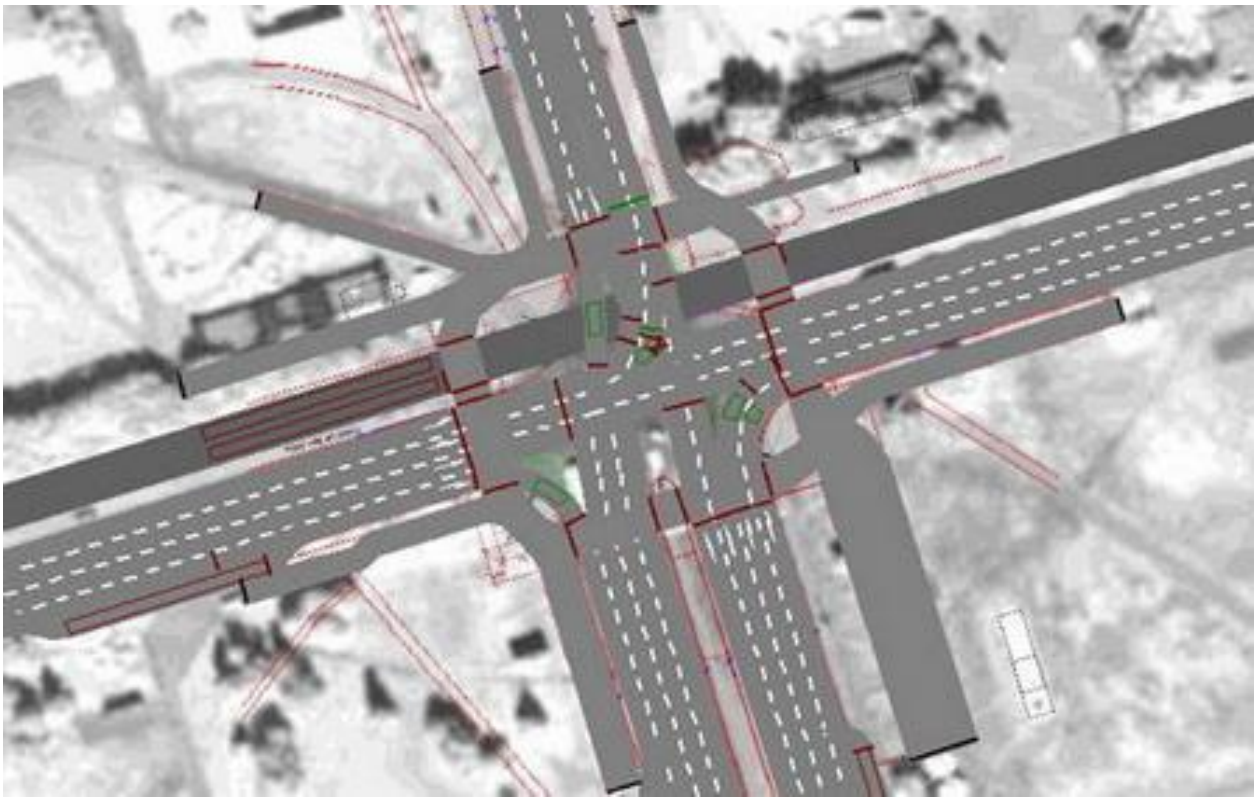


Рис. 3.5. Готовая модель перекрестка ул. Попова – ул. Коммунистическая

На сегодняшний день осуществляется реализация описанного выше проекта с учетом всех результатов моделирования.

3.5.3. Cистема Rapidis Traffic Analyst

Среди программных средств моделирования транспортного трафика стоит отметить продвигаемое *IBM* решение *Rapidis Traffic Analyst*, которое, по сути, представляет собой модуль расширения геоинформационного пакета *ArcGIS*.

ArcGIS – семейство программных продуктов американской компании *ESRI*, одного из лидеров мирового рынка геоинформационных систем. *ArcGIS*, построена на основе технологий *COM, .NET, Java, XML, SOAP*. Последняя версия – *ArcGIS* 9.3.

ArcGIS позволяет визуализировать, представить в виде цифровой карты большие объёмы статистической информации, имеющей географическую привязку. В среде создаются и редактируются карты всех масштабов: от планов земельных участков до карты мира. Также в *ArcGIS* встроен широкий инструментарий анализа пространственной информации.

Масштаб у *Traffic Analyst* шире, чем у *PTV*. Система позволяет проводить анализ на уровне регионов и страны в целом. В состав *Traffic Analyst* входит солидный набор инструментов для планирования движения и специализированных средств редактирования для работы с маршрутами и расписаниями общественного транспорта.

Все эти функции подключаются к среде обработки геопространственных данных *ArcGIS* Geoprocessing и встроенному конструктору моделей *ArcGIS*

Model Builder. Среди решаемых *Traffic Analyst* задач: моделирование колебаний спроса на перевозки, вызванных изменениями в инфраструктуре, демографии, политике и т. д., оценка последствий крупных инфраструктурных проектов, создание основы для оценки экологического влияния и, конечно же, прогнозирование транспортных потоков и анализа доступности транспортных средств.

3.5.4. Программа TRANSNET и модель транспортной системы Московской агломерации

Для компьютерной реализации транспортных моделей специалистами Института системного анализа РАН была разработанная программа *TRANSNET* [3.9]. *TRANSNET* представляет собой современное графическое *Windows*-приложение и по функциональным возможностям может быть охарактеризовано как интегрированная среда разработчика в области моделирования транспортных потоков.

TRANSNET включает в себя графический редактор транспортной сети, средства моделирования, средства представления и вывода результатов. Поскольку программное обеспечение проектировалось в первую очередь как средство разработчика, в него не «зашиты» фиксированные методики расчета потоков. Вместо этого обеспечено большое количество типичных расчетных алгоритмов и команд, из которых, как из кирпичиков, можно составлять самые разные расчетные схемы.

С использованием *TRANSNET* совместными усилиями специалистов Института системного анализа РАН, Центра исследований транспортной инфраструктуры и экспертов Российской академии архитектуры и строительных наук была создана модель транспортной системы Московской агломерации, охватывающая Москву с ближайшими пригородами. В модель включена УДС агломерации, метро и поезда пригородного сообщения. Развитая методика позволяет давать прогноз транспортных и пассажирских потоков в разное время суток (утренний час пик, средний дневной час, вечерний час пик). Модель откалибрована по имеющимся данным обследований и показывает хорошее совпадение расчетных показателей с реальными цифрами.

Основные применения модели связаны с прогнозом последствий принимаемых решений и выбором вариантов развития транспортной сети и территорий города. Модель дает ответ на вопрос, всегда стоящий перед планировщиками: какими станут транспортные потоки, и как изменятся характеристики движения, если провести модернизацию сети или построить новые жилые районы или другие объекты притяжения. Вместе с тем данная модель и программное обеспечение могут быть инструментом для исследования и анализа структурных проблем в транспортной сети. Например, при анализе потенциального спроса на те или иные участки сети, а также вынужденного перепробега из-за заторов. Другой пример – анализ эффективности системы маршрутов общественного транспорта.

Моделирование позволяет оценить спрос на пассажирские передвижения на отдельных дугах и сопоставить его с провозными способностями имеющихся маршрутов. Кроме того, моделирование позволяет оценить различные территории города с точки зрения транспортной доступности, что может иметь значение для оценки землепользования.

3.6. СРАВНЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ ОБЩЕСТВЕННОГО ТРАНСПОРТА

Наряду с первоочередными мероприятиями для повышения привлекательности общественного транспорта, такими как: обеспечение удобных взаимосвязей между районами города на общественном транспорте, оптимизация маршрутной сети наземного транспорта, рациональное распределение подвижного состава по маршрутам и так далее, многие исследователи выделяют важность задачи создания системы информирования пассажиров о режимах работы общественного транспорта [3.10].

Важное место среди программных средств, посвященных работе с транспортными потоками, занимают информационные системы, которые позволяют пользователям получать сведения о географическом расположении улиц города, зданий, остановок и работе общественного транспорта. Такие системы связаны с транспортной сетью определенного города и зачастую основаны на использовании электронных географических карт.

Ниже представлены несколько подобных систем, находящиеся в свободном доступе сети Internet.

3.6.1. Официальный сайт ЕМУП ТТУ

ЕМУП «Трамвайно-троллейбусное управление» сопровождает *web*-сайт *www.ettu.ru* [3.11]. Его основная функция – информирование граждан о работе данного предприятия. Сайт предоставляет следующие сведения о трамвайно-троллейбусном движении:

- трамвайные и троллейбусные маршруты, включая последовательность всех остановок в виде списка и в виде графического представления на карте города (см. рис. 3.6);

Маршрут № 3
«ЦПК и О – ВИЗ – ЦПК и О»(кольцевой)

- Депо: Западное
- Длина маршрута: 22,1 км.
- Время рейса: 100 мин.

- Путь следования: конечная станция "ЦПКиО" - ул. Мичурина - ул. Тверитина - ул. Луначарского - ул. Челюскинцев -ул. Московская - Дворец Молодёжи - Верх-Исетский бульвар - ул. Кирова - станция "ВИЗ" - ул. Красноуральская - ул. Татищева - ул. Викулова - пер. Тульский - ул. Волгоградская - ул. Белореченская - ул. Радищева - ул. 8 Марта - Цирк - ул. Куйбышева - ул. Луначарского - ул. Тверитина -ул. Мичурина - конечная станция "ЦПКиО".

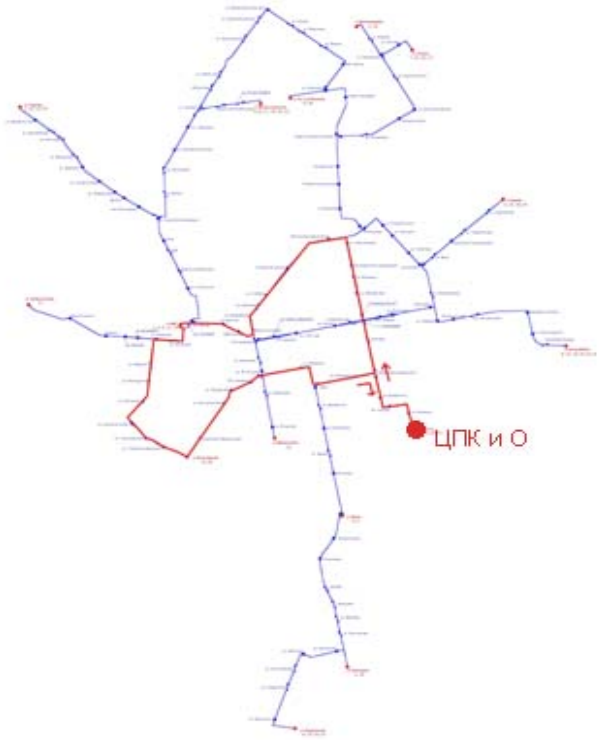


Рис. 3.6. Схема маршрута *www.ettu.ru*

- расписание движения трамваев и троллейбусов только через конечные остановки (см. рис. 3.7). Расписание меняется в зависимости от дня недели и в течение года.

Отправление с конечной станции

Часы	Минуты					Кол-во отправлений
6	20	30	53			3
7	04	14	26	40	52	5
8	03	19	31	42		4
9	08	18	30	40	54	5
10	07	18	28	43	54	5
11	05	21	32	43	53	5
12	07	19	31	43	58	5
13	08	18	45	56		4
14	19	43	57			3
15	10	32	46	57		4
16	12	23	35	47		4
17	13	24	36	47		4
18	01	06	23	39		4
19	02	15	30	40	52	5
20	02	18	37	53		4
21	11	24	35	43	57	5
22	07					1
23	22					1
Итого						71

Рис. 3.7. Расписание маршрута на сайте *www.ettu.ru*

Из выше сказанного очевидно, что информация, представленная на сайте, не позволяет пассажиру эффективно использовать общественный транспорт. Карта города имеет слишком мелкий масштаб и не дает четкого представления о маршрутах. Расписание движения по конечным остановкам фактически не имеет ценности для пассажиров, так как не дает представления о прохождении маршрутов через другие остановки города.

3.6.2. АИС «Транспорт города Екатеринбурга»

Автоматизированная информационная система «Транспорт города Екатеринбурга» представляет собой *web*-сайт *www.transport.ekburg.ru* [3.12], который в данный момент носит незавершенный характер. Многие заявленные сервисы до сих пор не реализованы. Основными целями разработки и внедрения автоматизированной информационной системы «Транспорт города Екатеринбурга» являются:

- обеспечение требуемого уровня информированности населения (полнота, точность, достоверность и своевременность предоставления информации);
- повышение эффективности и открытости муниципального управления;

- повышение эффективности использования информационных технологий и информационных ресурсов в городском хозяйстве;
- повышение эффективности взаимодействия с другими уровнями власти;
- создание механизмов взаимодействия с населением на базе новых информационных технологий.

Ниже приведены сведения об общественном транспорте, которые могут быть получены пользователями сайта на сегодняшний день:

- маршруты движения трамваев, троллейбусов, автобусов, метрополитена, маршрутного такси. При этом информация предоставляется только в виде последовательности остановок. Наблюдается некорректное отображение маршрутов на карте города, при увеличении масштаба рисунка появляется сообщение об ошибке;
- архив обо всех изменениях в маршрутах;
- информация о перевозчиках (общие сведения о предприятиях города; оказывающих услуги по общественным пассажироперевозкам);
- тарифы на проезд и льготные категории граждан;
- правила пользования общественным транспортом.

Кроме этого, на сайте заявлены такие полезные для пассажиров сервисы как «Оптимальный путь» и расписание движения транспорта. Но, к сожалению, уже на протяжении долгого времени, они остаются нереализованными, доступ к ним закрыт.

Из выше сказанного можно сделать вывод, что на сегодняшний день АИС «Транспорт города Екатеринбурга» предоставляет лишь общую информацию о дорожном движении в городе.

3.6.3. ДубльГИС

ДубльГИС – бесплатный электронный справочник организаций, объединенный с картой города. Отличное информационное наполнение, совместное использование данных справочника и карты, удобная система поиска помогают пользователям программы с легкостью ориентироваться в 17 городах России и Украины и находить нужную информацию. Существуют версии ДубльГИС для персонального компьютера, карманного персонального компьютера и онлайн-версия на сайте *map.2gis.ru*.

Электронная карта-справочник ДубльГИС для персонального компьютера – наиболее популярный продукт. Помимо различных видов поиска по карте и справочнику, в настольной версии предусмотрены возможности добавления собственной информации, печати данных, настройки внешнего вида, а также сервис ежемесячных обновлений.

ДубльГИС 3.0 – обновленная версия ДубльГИС для персонального компьютера, вышедшая в апреле 2008 года, характеризуемая новым современным интерфейсом, улучшенным функционалом, возможностями для разработчиков.

ДубльГИС 3.0 – это мощный инструмент, позволяющий решать большой спектр задач по поиску места на карте. Кроме этого, в приложении разработаны функции для работы с информацией об общественном транспорте.

Рассмотрим основные возможности ДубльГИС 3.0 для общественного транспорта на примере города Екатеринбурга.

Поиск маршрутов городского транспорта

При поиске маршрутов городского транспорта в справочнике «Транспорт» отображается список найденных маршрутов. Кликом из списка открывается карточка маршрута (рис. 3.8).

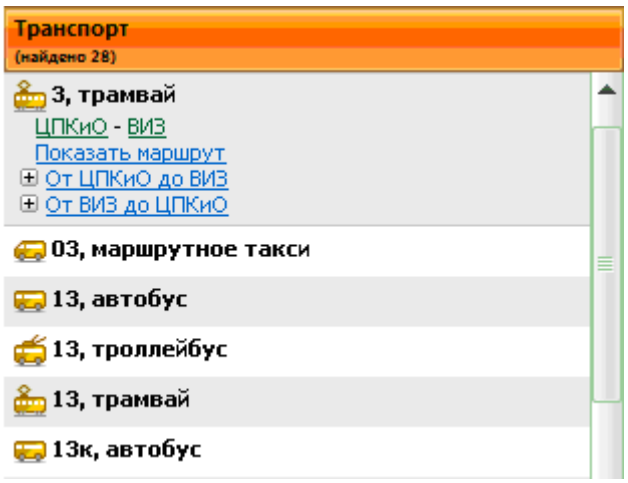


Рис. 3.8. Результат поиска маршрутов в ДубльГИС 3.0

В карточке отображается номер маршрута, тип транспорта (автобус, троллейбус, трамвай, маршрутное такси), конечные остановки, ссылка «Показать маршрут», позволяющая увидеть выбранный маршрут на карте в масштабе, при котором видны все остановки данного маршрута. Дополнительно в карточке маршрута расположены ссылки на списки остановочных пунктов от той или иной конечной остановки.

Кликнув мышкой на необходимую остановку выбранного маршрута, получим информационную карточку остановки.

Варианты проезда между остановками

При поиске проезда между необходимыми остановками в справочнике «Транспорт» отображается список вариантов проезда, в том числе с пересадками (см. рис. 3.9).

Содержание карточки проезда: начальная остановка, остановки пересадок, конечная остановка, количество пересадок, пеших переходов (в случае сложного варианта проезда).

При выборе ссылки «Показать на карте» можно увидеть на карте весь маршрут проезда. Если кликнуть по названию остановки или номеру маршрута, то на карте выделяются соответствующие остановки и маршруты городского транспорта. При клике на ссылку «Показать» (справа от номера маршрута) на карте отображается участок данного маршрута, который включен в предложенный вариант проезда.

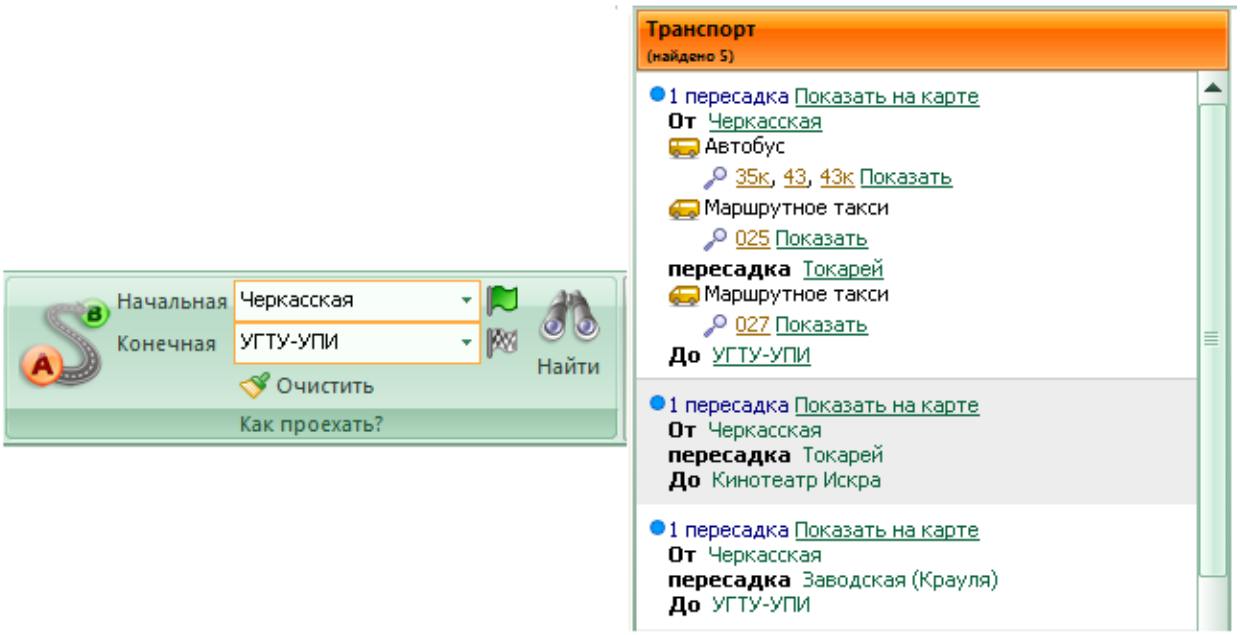


Рис. 3.9. Варианты проезда от ост. Черкасская до ост. УГТУ – УПИ

Преимущества ДубльГИС 3.0 для пассажиров общественного транспорта:

- очень удобное, наглядное представление информации о маршрутах на карте в любом масштабе;
- дружественный интерфейс, максимально приближенный к офисным приложениям *Windows Office* 2007;
- реализация поиска всех вариантов проезда между остановками;
- высокое быстродействие – поиск маршрутов и их прорисовка на карте осуществляется за очень короткое время.

Недостатки ДубльГИС 3.0 для пассажиров общественного транспорта:

- отсутствие информации о расписании и интервалах движения транспорта;
- отсутствие возможности сравнить найденные варианты проезда по какому-либо критерию оптимальности (время, расстояние, плата за проезд и прочее) и выбрать наиболее подходящий для каждого пользователя;
- ДубльГИС – это настольное приложение: выигрывая в быстродействии выполнения поиска, пользователь вынужден постоянно следить за актуальностью информации и периодически обновлять приложение.

3.6.4. Web-сайт «Маршруты Екатеринбурга»

Web-сайт «Маршруты Екатеринбурга», размещенный в сети *Internet* по адресу *http://eka.rusavtobus.ru*, представляет собой *we*b-приложение, основанное на использовании карт Google [3.13]. Карты *Google* – это служба *Google*, которая предлагает удобную для пользователя технологию поиска на карте и локальные данные о компаниях, включая адрес, контактную информацию и маршруты проезда [3.14]. Карты *Google* поддерживают такие уникальные функции, как интегрированные результаты поиска данных по компаниям, перетаскиваемые карты, спутниковые изображения, ландшафтные карты, просмотр улиц, подробные маршруты проезда и другие. Служба *Google* основана на технологии *web* 2.0, которая позволяет повернуть карту, приблизить или удалить ее с очень небольшой перерисовкой. Таким образом, работая с *web*-приложением в сети *Internet*, по быстродействию возникает ощущение, что работаешь с настольным приложением.

Сайт «Маршруты Екатеринбурга» предоставляет пользователям возможность найти все возможные маршруты между двумя точками на карте (рис. 3.10).

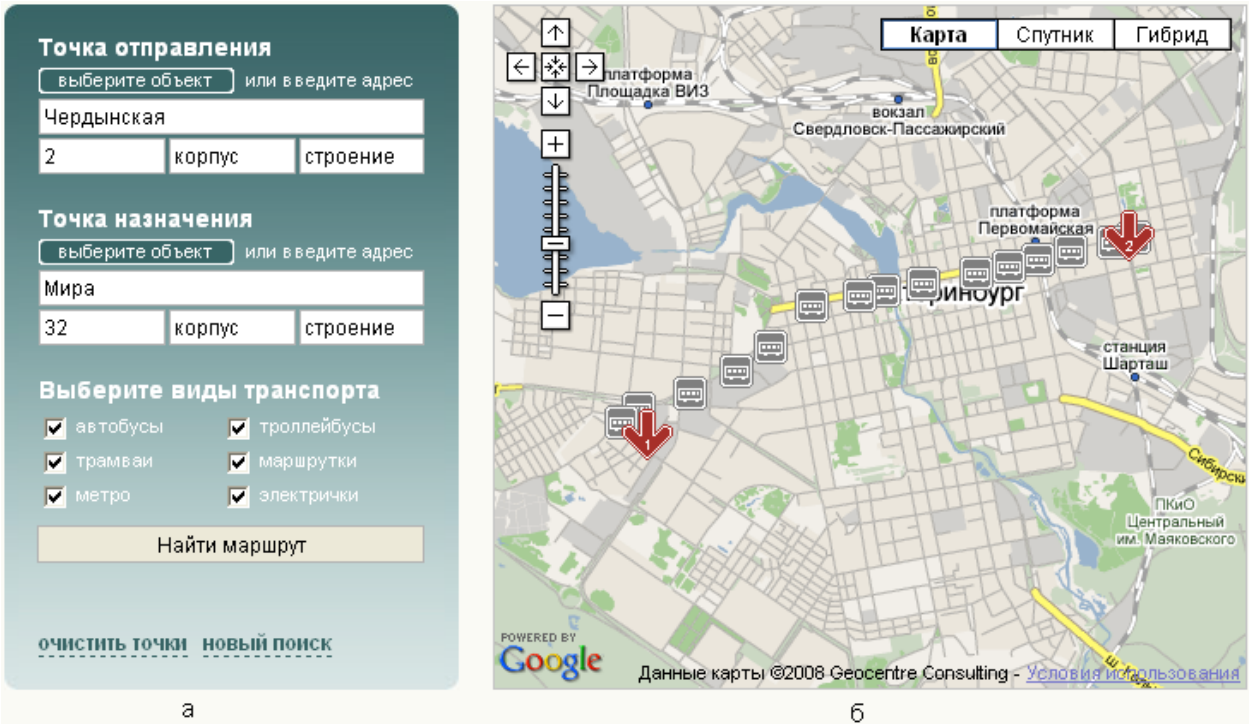


Рис. 3.10. Сайт «Маршруты Екатеринбурга»:
а) начальные параметры поиска;
б) графическое представление одного из найденных маршрутов на карте

В качестве результата выдаются все возможные маршруты и их параметры:

- графическое представление маршрута на карте *Google* (рис. 3.10);
- расчетное время в пути (см. рис. 3.11);
- остановки и маршруты всех пересадок, а также их общее количество;

- расстояние, которое необходимо преодолеть пешком;
- стоимость проезда за весь маршрут;
- количество попутчиков – пользователей системы, получивших такой же маршрут по своему запросу.

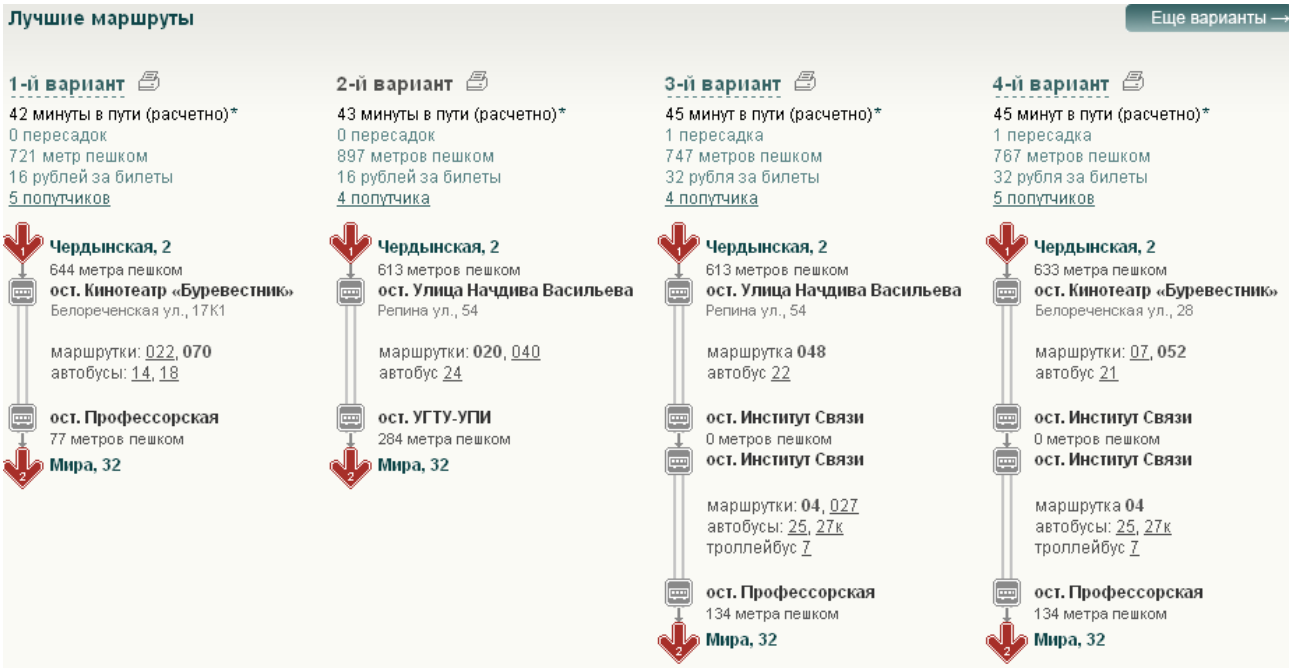


Рис. 3.11. Результаты поиска маршрутов

Преимущества сайта «Маршруты Екатеринбурга» для пассажиров:

- наглядное представление информации о маршрутах на карте в любом масштабе без длительного ожидания прорисовки;
- возможность найти множество маршрутов и сравнить их по некоторым критериям;
- возможность задать начальные и конечные пункты адресами или местом расположения на карте;
- возможность задать виды транспорта, которые следует учитывать при построении маршрута;
- свободный доступ к сервису в сети *Internet*, без необходимости следить за обновлением приложения.

Основные недостатки системы «Маршруты Екатеринбурга» связаны с субъективностью результатов поиска маршрутов и их характеристик. Главная причина этого – допущения, принятые на сайте. «Время в пути рассчитывается, исходя из скорости движения метро и электричек равной 45 км/ч, скорости маршрутных такси, равной 35 км/ч, скорости наземного транспорта, равной 30 км/ч, скорости пешехода – 4 км/ч плюс время на пересадку, которое зависит от видов транспорта и от расстояния, которое надо пройти пешком. Если на одном отрезке предлагаются виды транспорта с разной скоростью – в

статистике маршрута указывается наилучшее время в пути». Таким образом, в системе не учитывается время отправления пассажира из начального пункта, расписание движения транспорта и его зависимость от времени суток,

Следствием этого являются некорректно рассчитанные характеристики маршрутов:

- время в пути (не учтено ожидание на остановках, зависящее от расписания маршрутов и интервалов их движения в течение суток);
- количество попутчиков (не может быть использовано как характеристика загруженности маршрутов, так как при поиске маршрутов не учитывается время отправления пассажира из начального пункта).

Другой проблемой сайта «Маршруты Екатеринбурга» является несоответствие предоставляемой информации реальным условиям транспортной сети города Екатеринбурга. При работе с сайтом были обнаружены следующие ошибки:

- неактуальная информация о маршрутах города. Например, при поиске маршрута «от: Чердынская, 2 – до: Мира, 2» (рис 3.11) во втором варианте пути присутствует маршрутка 020, которая на данный момент отменена. Также ни в одном из найденных вариантов не участвует автобус 37, маршрут которого проходит через остановку Черкасскую, расположенную в непосредственной близости от пункта отправления. Подобные ошибки встречаются достаточно часто и при поиске других маршрутов;
- неверный подсчет стоимости проезда по всему пути. Например, в первом найденном варианте (рис. 3.11) указана стоимость проезда 16 руб. При выборе любого из предложенных маршрутов стоимость проезда будет отличаться от найденной: на маршрутках 070, 022 – 10 руб., на автобусах 14, 18 – 11 руб.

Исследовав работу сайта «Маршруты Екатеринбурга», можно прийти к выводу: пользователь сайта получает только приближенную информацию о некоторых существующих маршрутах между пунктом отправления и назначения. Характеристики найденных маршрутов (расчетное время в пути, стоимость проезда) не могут быть объективными критериями для сравнения их оптимальности, так как содержат ошибки и большую долю вероятности.

3.6.5. Информационная система транспорта Лондона

Transport for London – интернет-портал [3.15], представляющий собой информационную систему общественного транспорта Лондона (*www.tfl.gov.uk*). Помимо большого количества важной информации о городском транспорте, портал предоставляет пользователям сервис Journey Planner (планировщик поездок). С помощью данного сервиса пользователь может получить информацию обо всех возможных маршрутах из одной точки Лондона в другую и при помощи конфигураторов поиска выбрать наиболее подходящий для него. При поиске пути, кроме заданных пользователем условий, учитываются

расписание движения транспорта и текущая дорожная ситуация (ремонтные работы, временное перекрытие улиц и прочее). Для удобства иностранных пользователей сайта, сервис доступен на многих языках мира, в том числе на русском.

Рассмотрим использование сервиса *Journey Planner* на примере поиска пути от станции *Euston* до станции *Holborn*.

1. Задание параметров поиска показано на рис. 3.12. В качестве параметров поиска могут быть использованы остановки общественного транспорта, адреса, названия достопримечательностей, учреждений и прочее.

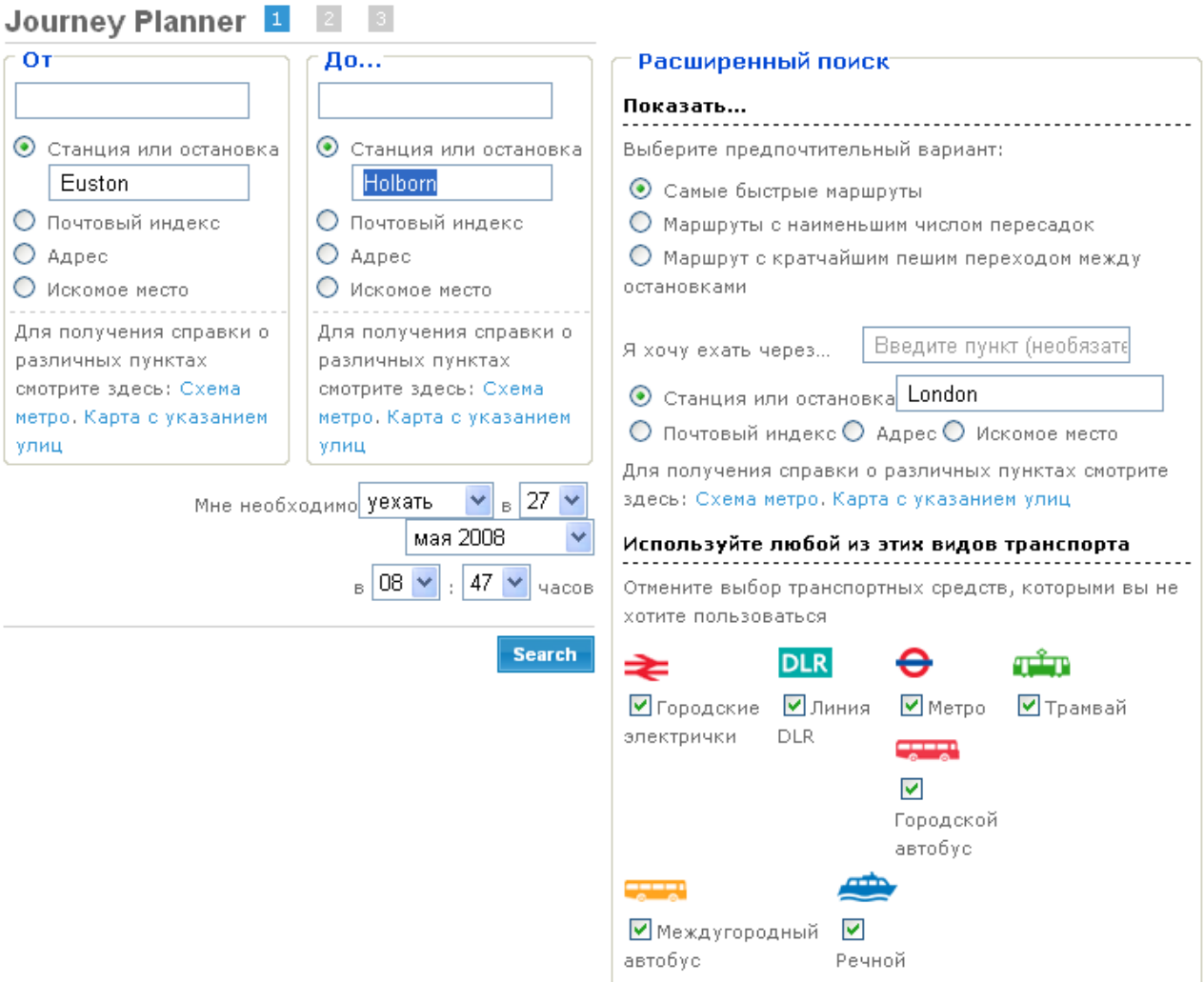


Рис. 3.12. Задание параметров поиска маршрутов

2. Результаты поиска показаны на рис. 3.13.

3. Существует возможность детального просмотра каждого варианта маршрута. На рис. 3.14 показан первый из найденных выше вариантов.

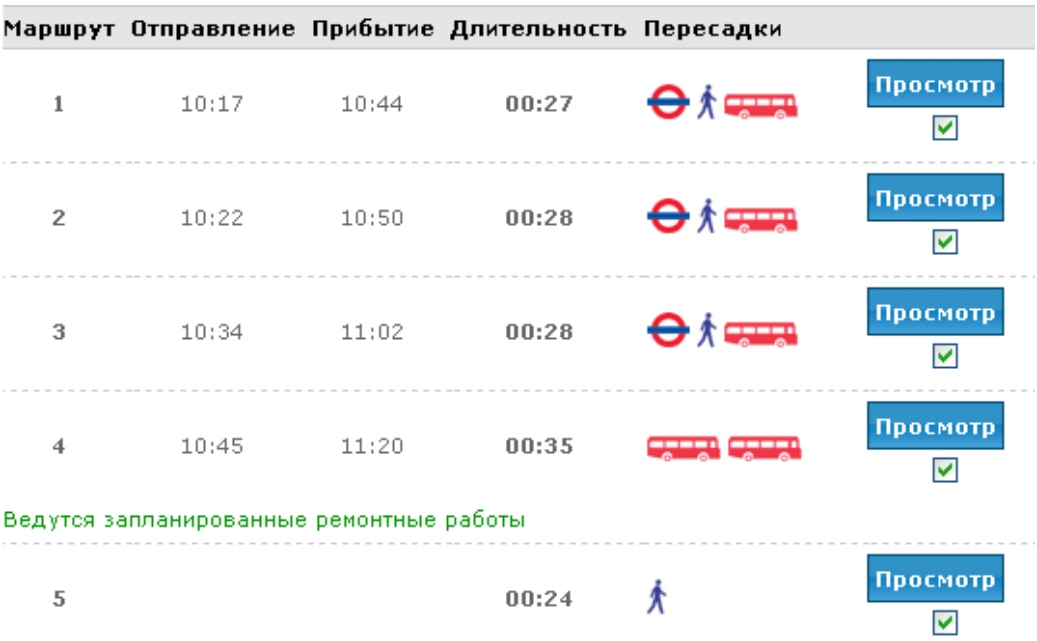


Рис. 3.13. Варианты маршрутов, найденные по запросу

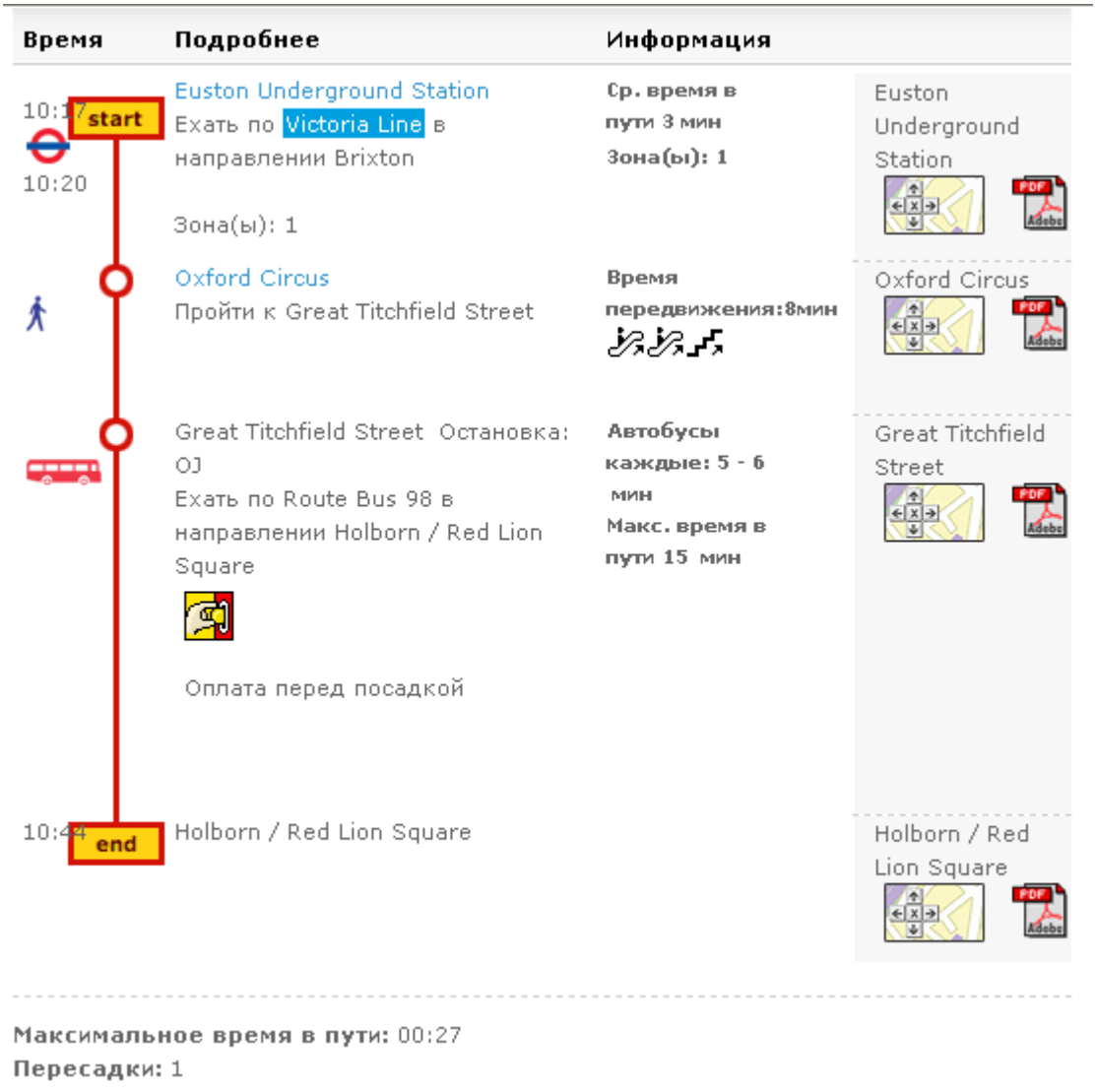


Рис. 3.14. Детальный просмотр найденного маршрута

С помощью описанного выше сервиса пользователь не только получает достоверную информацию обо всех возможных вариантах маршрута из одной точки города в другую, но и может сконфигурировать наиболее удобный для себя маршрут, учитывая предпочтения в используемом виде транспорта, материальные и физические возможности.

3.6.6. Сравнительная таблица возможностей рассмотренных ИС

Сравнительный анализ описанных выше информационных систем общественного транспорта по их функциональным возможностям приведен в табл. 3.1.

Таблица 3.1

Сравнительный анализ ИС общественного транспорта

Критерии сравнения	Сайт ЕМУП ТТУ	Сайт АИС «Транспорт города Екатеринбурга»	Электронный справочник «Дубль-ГИС»	Сайт «Маршруты Екатеринбурга»	Сайт «Transport for London»	Авторская учебная модель для абстрактного города
Способ реализации и доступность пользователям	Интернет-сайт со свободным доступом	Интернет-сайт со свободным доступом	Бесплатное настольное приложение с обновлениями	Интернет-сайт со свободным доступом	Интернет-сайт со свободным доступом	Настольное приложение
Виды транспорта, по которым предоставляется информация	Трамвай, троллейбус	Трамвай, троллейбус, автобус, маршрутки, метро	Трамвай, троллейбус, автобус, маршрутки	Трамвай, троллейбус, автобус, маршрутки, метро	Все виды ГПТ Лондона	Нет разделения по видам
Способ представления маршрута	Список остановок + графически на карте без возможности увеличения	Список остановок	Список остановок + графически на электронной карте с навигацией	Графически на карте Google	Список остановок + графически на электронной карте с навигацией	Список остановок + графически на карте
Информация о расписании движения ГПТ	Только для конечных станций	Сервис заявлен, но не реализован	Нет	Нет	Информация об интервалах движения	Расписание маршрутов по каждой остановке

Окончание табл. 3.1

Критерии сравнения	Сайт ЕМУП ТТУ	Сайт АИС «Транспорт города Екатеринбурга»	Электронный справочник «Дубль-ГИС»	Сайт «Маршруты Екатеринбурга»	Сайт «Transport for London»	Авторская учебная модель для абстрактного города
Поиск маршрутов с пересадками между двумя точками	Нет	Сервис заявлен, но не реализован	Есть	Есть	Есть	Есть
Возможность сравнения маршрутов по каким-либо критериям и выбора оптимального маршрута	Нет	Нет	Нет	Сравнение некорректно из-за субъективности характеристик и ошибок	Можно выбирать пути по множеству критериев, в том числе по оптимальному времени	Поиск оптимального по времени пути, анализ зависимости пути от условий поиска, сравнение
Учет дорожной ситуации (ремонт дороги, изменение маршрута и др.)	Нет	Нет	Нет	Нет	Информация о маршрутах выдается с сообщением о закрытых участках	Учет закрытых участков при поиске маршрутов
Учет пробок	Нет	Нет	Нет	Нет	Нет	Нет
Резюме: удобство и эффективность использования для пассажиров	Предоставление общей информации о ГПТ, неудобный интерфейс, не имеет большой ценности для пассажиров	Низкая информативность, многие сервисы не реализованы, сайт на незавершенной стадии не эффективен	Удобный интерфейс, высокая скорость и информативность, высокий эффект от пользования	Информация не может быть использована для объективной оценки маршрутов	Высокая информативность и гибкость, возможность конфигурации маршрутов по множеству параметров	Цель – создать прототип ИС ГПТ, сочетающей удобство и высокую информативность при выборе маршрута для пассажира

Из сравнительного анализа видно, что ни одна из существующих систем не предоставляет жителям города Екатеринбурга полной информации,

необходимой для комфортного и эффективного использования ГПТ. В том числе, для нашего города не найдено ни одного решения, позволяющего находить оптимальный по времени путь из одной точки нашего города в другую с учетом времени старта и дорожной ситуации.

Внедрение подобной системы имело бы большой социально-экономический эффект для любого крупного города России, в том числе для Екатеринбурга. Прототипом такой системы является разработанная модель, о чем свидетельствует сравнительная табл. 3.1.

3.7. ПРОБЛЕМА УЧЕТА ДОРОЖНОЙ СИТУАЦИИ В ИС ГПТ

В настоящее время увеличение количества транспортных средств как личных, так и общественных, привело к перегруженности городских дорог, многочасовым пробкам, увеличению количества аварий и дорожно-ремонтных работ.

Все эти факторы негативно влияют на работу ГПТ. Каждый житель Екатеринбурга знает, что автомобильная пробка или ремонт дороги на центральных улицах парализуют движение автобусов, троллейбусов и даже трамваев. Например, при свободном движении путь из центра города до УГТУ – УПИ на трамвае занимает 15 – 20 минут. При образовании заторов на улице Ленина, перекрестках улиц Ленина – Луначарского и Ленина – Восточная этот же путь может занять около часа.

Возникает вопрос учета дорожной ситуации при информировании пассажиров о работе ГПТ. Пример учета таких ситуаций, как ремонт и перекрытие участков дорог, временное изменение маршрутов, показан в п. 3.6.5. В ИС «*Transport for London*» по запросу пользователя о каком-либо маршруте система выдает предупреждение о тех участках дорог, которые закрыты или затруднены для проезда (см. рис. 3.13).

Подобные ситуации строго регламентированы соответствующими государственными структурами, информация о них поступает в транспортные предприятия и средства массовой информации, поэтому их учет в информационных системах ГПТ вполне реален.

Хотя большинство существующих ИС общественного транспорта не учитывают дорожную ситуацию (табл. 3.1), в разрабатываемой модели это требование будет учтено.

Гораздо сложнее вопрос об учете автомобильных пробок и заторов, которые сами по себе уже давно стали объектом научных исследований. Автомобильные пробки – это пример «трагедии ресурсов общего пользования» [3.16]. У водителей нет причин не использовать дорожные ресурсы сверх меры, вплоть до их полного исчерпания и наступления кризиса. И пока автоконцерны разрабатывают варианты миниатюрных авто, а городские власти ищут пути сокращения количества автомобильных заторов путем разгрузки дорог, американский политолог и экономист Энтони Даунс (*Anthony Downs*) вывел

следующий эмпирический закон: «технологические решения не помогут снизить число пробок» [3.17]. Даунс рассматривает автомобильные пробки как следствие одного из правил рыночной экономики. Увеличение пропускной способности дорог, по его мнению, приведет лишь к увеличению «спроса» на них у большего числа автомобилистов, поэтому любое самое широкое шоссе рано или поздно «встанет».

Таким образом, автомобильные пробки – это объективная реальность, которая вносит свои коррективы в жизнь каждого жителя крупного города. Чтобы минимизировать это негативное влияние необходимо иметь информацию о ситуации на дорогах. В настоящее время существует множество сервисов, предоставляющих информацию о загруженности улиц. Основные источники данных для систем мониторинга – камеры, автоматические видеодетекторы движения, расположенные на городских улицах, снимки с космических спутников, корреспонденты собственных служб. Достоверность информации оценивается в $50-95\,\%$ в зависимости от участка дороги. Практически все подобные средства информирования о пробках – сервисы сотовых операторов, программное обеспечение для карманных персональных компьютеров и персональных навигационных систем – являются платными. Поисковая система «Яндекс» поддерживает бесплатный вариант сервиса «Яндекс.Пробки».

Особенностью всех подобных сервисов является то, что они лишь предоставляют информацию о ситуации на дорогах в режиме реального времени. Ни одна система не имеет возможности с достаточно высокой точностью прогнозировать автомобильные пробки и их поведение даже на малые периоды. Это связано со следующими свойствами транспортных потоков:

- стохастичность транспортных потоков: их характеристики допускают прогноз только с определенной вероятностью, так как напрямую зависят от таких случайных факторов, как погодные условия, ДТП и т.п.;
- нестационарность транспортных потоков, причем колебания их характеристик происходят как минимум в трех циклах: суточном, недельном и сезонном;
- неполная управляемость, суть которой состоит в том, что даже при наличии полной информации о потоках и возможности информирования водителей о необходимых действиях, эти требования носят рекомендательный характер. Поведение каждого отдельного водителя непредсказуемо.

Из вышесказанного следует, что автомобильные пробки не могут быть корректно учтены в информационных системах общественного транспорта, в том числе при предоставлении информации о расписании движения и времени, требуемом на преодоление какого-либо пути. Обоснуем данное утверждение.

Допустим, что в информационной системе заявлен сервис учета пробок и текущей загруженности дорог. Пользователь посылает запрос в систему на поиск оптимального по времени пути из пункта А в пункт В. Система из каких-либо источников получает абсолютно достоверную информацию о текущей дорожной ситуации. Пусть в системе существует методика анализа характеристик транспортного потока, позволяющая прогнозировать время передвижения транспорта в заданных условиях. Используя эти расчеты, система выдает оптимальный по времени маршрут.

Приведем некоторые обстоятельства, доказывающие несостоятельность найденного решения.

1. Запрос может обрабатываться только с учетом текущей ситуации. Даже через несколько минут ситуация может измениться так, что просчитанный ранее маршрут перестанет быть оптимальным. Причина кроется в уже упомянутой стохастичности транспортных потоков. Резкое изменение погоды, авария на дороге, проезд длинной автоколонны, сломанный светофор – все события не могут быть заранее спрогнозированы и просчитаны. Поэтому результат запроса, даже с учетом дорожной ситуации на момент его поступления, может оказаться недостоверным.

2. Вне зависимости от поведения транспортного потока общественный транспорт имеет утвержденное расписание. При всей нестабильности дорожной ситуации водители составов и диспетчеры прикладывают максимум усилий, чтобы минимизировать отклонения от предписанного времени. Существуют меры, позволяющие даже в сложных условиях поддерживать максимально возможное соблюдение расписания: выпуск дополнительных составов, сокращение стоянок на конечных станциях и прочее. Эти меры, в совокупности с человеческим фактором, не могут быть учтены ни одной системой.

3. Как показано в п. 3.5, методики, позволяющие с высокой точностью прогнозировать время передвижения транспорта в условиях транспортного потока, очень сложны и включают в себя длительный и трудоемкий процесс моделирования, комплексные расчеты и анализ. Использование таких методик в информационных системах общественного транспорта маловероятно и неоправданно.

Таким образом, учет автомобильных пробок в информационных системах общественного транспорта нецелесообразен и не ведет к уменьшению роли случайного фактора в предоставляемых результатах запросов. Чтобы повысить достоверность информации о работе ГПТ, а вместе с ней и удобство его использования, необходимо обеспечить соблюдение расписания. А для этого следует на самом высоком уровне признать абсолютный приоритет общественного транспорта в городских пассажирских перевозках. Должны быть предусмотрены такие меры, как обособление полос для движения ГПТ, корректировка устаревших статей Правил дорожного движения, их выполнение на улицах города и многое другое.

3.8. РЕЗУЛЬТАТЫ ОБЗОРА СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В ходе обзора предметной области были рассмотрены основные научные и технические подходы к моделированию транспортных потоков и их исследованию.

Было показано, что проблема оптимизации дорожного движения, в том числе движения общественного транспорта стала актуальна уже давно. С ростом городского населения и повсеместной автомобилизации эта проблема приобретает новые более сложные аспекты, занимая все более значимое место в списке остро стоящих вопросов улучшения городской инфраструктуры. Как справедливо отметил бывший заместитель министра транспорта РФ Александр Колик, «доступность и качество городского транспорта во многом определяют и реальный уровень жизни, и социальный климат, и мнение людей об эффективности органов власти» [3.18].

Существует множество научных исследований, посвященных проблеме транспортных потоков. На их основе разработаны модели, системы и их программные реализации различной точности и сложности. По решаемым задачам эти средства можно разделить на две большие группы.

Первая группа включает в себя системы, реализующие моделирование транспортных сетей и потоков, их анализ и оценку, планирование транспортной инфраструктуры и общественного транспорта, транспортных сетей; прогноз запланированных мероприятий, создание платформы для транспортно-информационных систем. Эти системы, как показано в п. 3.5, являются серьезными коммерческими продуктами, которые разрабатываются и поддерживаются на протяжении долгого времени научными институтами и проектными организациями.

Вторая группа включает в себя информационные системы транспорта, основной целью которых является информирование населения о дорожном движении и работе ГПТ определенной транспортной сети. Эти системы обычно реализованы в виде web-сайтов или постоянно обновляемых настольных приложений и зачастую основаны на использовании электронных географических карт.

Попробуем объединить в себе задачи обеих групп в упрощенной форме.

С одной стороны, конечная цель – это разработка информационных сервисов для пассажиров ГПТ, в том числе поиск оптимальных маршрутов в заданной транспортной сети с фиксированным расписанием. При этом считается, что сама сеть, маршруты и расписание являются заданными и не подлежат изменению.

С другой стороны, на начальном этапе полезно создать настольную модель сети общественного транспорта, приближенную к реальным сетям крупных Российских городов. На этой модели проверить и отработать алгоритмы разрабатываемых сервисов, изучить дополнительные возможности по анализу и оценке эффективности сети общественного транспорта.

3.9. МОДЕЛИРОВАНИЕ И ОПТИМИЗАЦИЯ МАРШРУТА ГОРОДСКОГО ЭЛЕКТРОТРАНСПОРТА ДЛЯ ПАССАЖИРА

Представлена авторская модель городского транспорта для абстрактного города[1]. Размер сети, количество маршрутов произвольны. Приводится математическая модель сети, описывается алгоритм поиска оптимального по времени проезда от одной остановки до другой. Построена программная реализация модели и алгоритма.

Каждый день большое количество жителей города пользуется общественным транспортом и зачастую вынуждены тратить слишком много времени на ожидание нужного маршрута. Кроме того, с ростом города происходит увеличение транспортной сети, появляются новые маршруты и остановки. Пассажиры из-за недостатка информации вынуждены самостоятельно составлять свой маршрут, совершая при этом ненужные пересадки, затрачивая не только лишние деньги на проезд, но и время.

Очевидно, что в настоящее время многие эти проблемы можно решить внедрением информационно-коммуникационной системы, предназначенной для информирования населения города о работе общественного транспорта.

В рамках городской целевой программы «Электронный Екатеринбург», утвержденной Екатеринбургской городской Думой, внедряется программный комплекс «Транспорт города Екатеринбурга» в виде информационного сайта. Организатором является администрация города Екатеринбурга, в лице Комитета по промышленности, науке, связи и информационным технологиям и Комитета по транспорту и организации дорожного движения. В настоящее время выполнена интеграция сайта с системой АСУ ЕМУП ТТУ.

Основной целью сайта ЕМУП ТТУ «Транспорт города Екатеринбурга» является обеспечение населения информацией о маршрутах трамваев и троллейбусов. Одним из востребованных сервисов является поиск оптимального пути для проезда пассажира из одного пункта в другой с пересадками. Данный сервис на сайте ЕМУП ТТУ пока не реализован. Сайт находится в открытом доступе и предназначен для всех пользователей сети Интернет: *http://transport.myitems.ru/, http://ettu.ru/*. Во многих странах мира внедрены информационные системы городского транспорта. Например, сайт *http://www.tfl.gov.uk/* представляет собой ИС транспорта Лондона. Система реализует подбор оптимального маршрута в соответствии с рядом критериев.

Разработано авторское приложение, моделирующее транспортную сеть и расписание движения транспорта абстрактного города. Данная модель используется для тестирования алгоритмов и оценки их трудоемкости. В модели предусмотрено: создание прямоугольной системы улиц в городе произвольного размера; создание заданного количества произвольных прямолинейных и круговых маршрутов; задание расписания для каждого маршрута.

[1] Работа выполнена в рамках дипломного проекта Низовой М.Ю. под руководством Трофимова С.П.

Приложение строит оптимальный маршрут от одной остановки до другой при заданном времени старта пассажира из начального пункта. При этом показываются все пересадки в пути, время, затраченное в пути и на ожидание транспорта.

Генерация необходимых данных производится с использованием встроенного в среду разработки генератора случайных чисел. Параметры распределений подобраны так, чтобы результаты оказались максимально приближены к реальной транспортной сети и соответствовали начальным пользовательским данным.

Работа с базой данных, в том числе создание и заполнение всех таблиц, осуществляется через *SQL*-запросы. В качестве механизма доступа к базе данных из приложения используется механизм *ADO*.

3.10. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ТРАНСПОРТНОЙ СЕТИ

Построим множество X всех остановок, пронумерованных натуральными числами.

Перегоном p_{ij} назовем участок дороги между остановками i, j, не содержащий других остановок. Перегон является направленным участком. Множество всех имеющихся перегонов обозначим P.

Пронумеруем трамвайные маршруты также натуральными числами n.

Траекторией маршрута n назовем последовательность связанных перегонов

$$T(n) = \{ p_{i_1 i_2}, ..., p_{i_{m(n)-1} i_{m(n)}} \}, \tag{3.1}$$

где $m(n)$ – количество остановок в маршруте n, включая начальную и конечную остановки. Траектории могут содержать повторяющиеся перегоны.

Если вагон маршрута n вышел с начальной остановки в момент времени t_{start}, то моменты его прибытия на очередные остановки задаются временной последовательностью

$$V(n, t_{start}) = \{ t_1 = t_{start}, t_2(n, t_{start}), ..., t_{m(n)}(n, t_{start}) \}. \tag{3.2}$$

Назовем графиком маршрута совокупность траектории маршрута и временной последовательности маршрута

$$G(n, t_{start}) = (T(n),\ V(n, t_{start})). \tag{3.3}$$

Маршрут может иметь несколько графиков движения в пределах суток.

3.10.1. Алгоритм моделирования транспортной сети

С учетом математической модели (3.1) – (3.3), моделирование транспортной сети требует задания следующих множеств:

- множества всех остановок транспортной сети;

- множества траекторий маршрутов;
- множества графиков движения маршрутов.

Моделирование каждого из множеств осуществляется:

- генерацией данных с учетом заданных пользователем параметров;
- путем обработки полученных данных и занесения их в базу данных с помощью *SQL*-запросов.

Подробный алгоритм моделирования изложен ниже:

1. Пользователь вводит начальные параметры абстрактного города:

a) M – количество улиц по горизонтали в прямоугольной системе города;

b) N – количество улиц по вертикали в прямоугольной системе города;

c) L – количество прямолинейных маршрутов в городе;

d) C – количество круговых маршрутов в городе.

2. В таблицу «Остановки» БД заносятся m×n перекрестков, их координаты и названия, привязанные к географическому положению.

3. Создание прямолинейных маршрутов. Для каждого из L прямолинейных маршрутов:

a) выбирается случайным образом ($x0$, $y0$) – начальная точка маршрута;

b) в зависимости от расположения ($x0$, $y0$) относительно диагоналей прямоугольника, образованного перекрестками города, выбирается направление движения маршрута;

c) случайным образом выбирается количество поворотов в маршруте – опорных точек маршрута. Их количество ограничено величиной, зависящей от размеров города;

d) последовательным связыванием опорных точек маршрута определяются все остальные остановки в маршруте и их последовательность – как все перекрестки, встречающиеся на пути связывания опорных точек;

e) в таблицу «Маршруты» заносится название маршрута, соответствующее порядковому номеру создания, и количество остановок в маршруте;

f) в таблицу «Конфигурации маршрутов» заносится последовательность остановок, по которым проходит маршрут, а именно: номер маршрута, номер остановки и её порядковый номер в соответствующем маршруте.

4. Создание круговых маршрутов. Для каждого из C прямолинейных маршрутов:

a) выбирается случайным образом (xc, yc) – точка, задающая центр окружности, по которой следует маршрут;

b) случайным образом выбирается радиус окружности, по которой следует маршрут. Его длина ограничена величиной, зависящей от размеров города;

c) находятся координаты всех перекрестков, лежащих на окружности, по которой следует маршрут. Эти точки являются опорными для маршрута;

d) последовательным связыванием опорных точек маршрута, определяются все остальные остановки в маршруте и их последовательность – как все перекрестки, встречающиеся на пути связывания опорных точек;

e) последней добавляется еще одна опорная точка, равная первой опорной точке. Таким образом, маршрут становится круговым;

f) в таблицу «Маршруты» заносится название маршрута, соответствующее порядковому номеру создания, и количество остановок в маршруте;

g) в таблицу «Конфигурации маршрутов» заносится последовательность остановок, по которым проходит маршрут, а именно: номер маршрута, номер остановки и её порядковый номер в соответствующем маршруте.

5. В таблицу «Виды расписания» заносится вид, соответствующий рабочим дням. Для упрощения моделирования это единственный вид расписания и все записи таблицы «Расписание» соответствуют именно ему.

6. Создание расписания. Для каждого из $L + C$ маршрутов:

a) создается базовое расписание – как последовательность времени прибытия на каждую из остановок маршрута. Базовое расписание совпадает с расписанием первого состава, вышедшего по этому маршруту в рабочий день;

b) создается последовательность интервалов времени, через которые следуют составы по данному маршруту;

c) с помощью базового расписания и интервалов следования составов создается полное расписание маршрута – все времена прибытия данного маршрута на каждую из остановок. Соответствующие записи заносятся в таблицу «Расписание».

В результате этих действий создается база данных, хранящая информацию об абстрактном городе, позволяющая воспроизвести модель его транспортной сети, а именно БД содержит:

- карту города – множество перекрестков;
- карта всех маршрутов города;
- расписание всех маршрутов города.

3.10.2. Программная реализация моделирования транспортной сети

При моделировании транспортной сети и расписания абстрактного города используются две основные операции.

1. Генерация данных с учетом задаваемых пользователем параметров.

2. Обработка полученных данных и занесение их в базу данных с помощью *SQL*-запросов.

Генерация необходимых данных производится с использованием встроенного в среду разработки генератора случайных чисел. Параметры датчика выбираются так, чтобы результаты оказались максимально приближены к реальной транспортной сети и соответствовали начальным пользовательским данным.

Работа с базой данных, в том числе создание и заполнение всех таблиц, осуществляется через *SQL*-запросы.

В качестве механизма доступа к базе данных из приложения был выбран механизм *ADO*. *ADO* (*Active Data Objects*) – это высокоуровневый компонент технологии доступа к данным от *Microsoft*. *ADO* – это программный интерфейс к любым типам данных, включая реляционные базы данных.

Компонент *TADOQuery* обеспечивает применение *SQL*-запросов при работе с данными через *ADO*. По своей функциональности он подобен стандартному компоненту запроса.

Каждый из шагов алгоритма реализован в виде пары функций, одна из которых генерирует необходимые данные, вторая осуществляет связь с базой данных и запись информации в соответствующую таблицу.

3.11. АЛГОРИТМ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОИСКА ОПТИМАЛЬНОГО МАРШРУТА

В работе использованы идеи, изложенные в статье [3.19].

Допустим, на остановке A в начальный момент времени $t_{\text{нач}}$ находится пассажир, которому необходимо приехать на остановку B за минимальное время. Путь пассажира может содержать пересадки.

Рассмотрим алгоритм решения данной задачи, основанный на принципе динамического программирования.

Назовем меткой $M(i)$ произвольной остановки i совокупность графика и времени прибытия на данную остановку по данному графику.

$$M(i) = (G(n, t_{start}), t_i(n, t_{start})) .$$

Метка остановки позволяет определить, на каком маршруте и из какой предыдущей остановки мы прибыли на остановку i. Метка с более ранним временем прибытия затирает прежнюю метку. Метки позволяют восстановить путь пассажира в обратную сторону: от остановки i до начальной остановки A. Остановка A метки не имеет.

Расстановка меток остановок осуществляется по волновому принципу: сначала рассматриваются остановки, находящиеся от A на расстоянии радиуса, равного одному перегону, затем двум перегонам и т.д.

Критерий остановки алгоритма – при просмотре всех остановок очередного радиуса ни одна из меток не изменилась.

Если конечная остановка *B* стала помеченной со временем приезда $t_{кон}$, то просмотр перегонов, исходящих из других помеченных остановок с большим, чем $t_{кон}$ временем, не производится.

Допустим, при увеличении радиуса мы пришли на некоторую помеченную остановку *i*, и время новой метки хуже, чем время прежней метки *M*(*i*) этой остановки. Тогда, очевидно мы сохраняем метку *M*(*i*) и дальнейшее увеличение радиуса через остановку *i* не производим.

Таким образом, можно уменьшить трудоемкость алгоритма.

Приложение написано в интегрированной среде разработки *Borland C++ Builder*.

Программная реализация алгоритма включает в себя две рекурсивные функции.

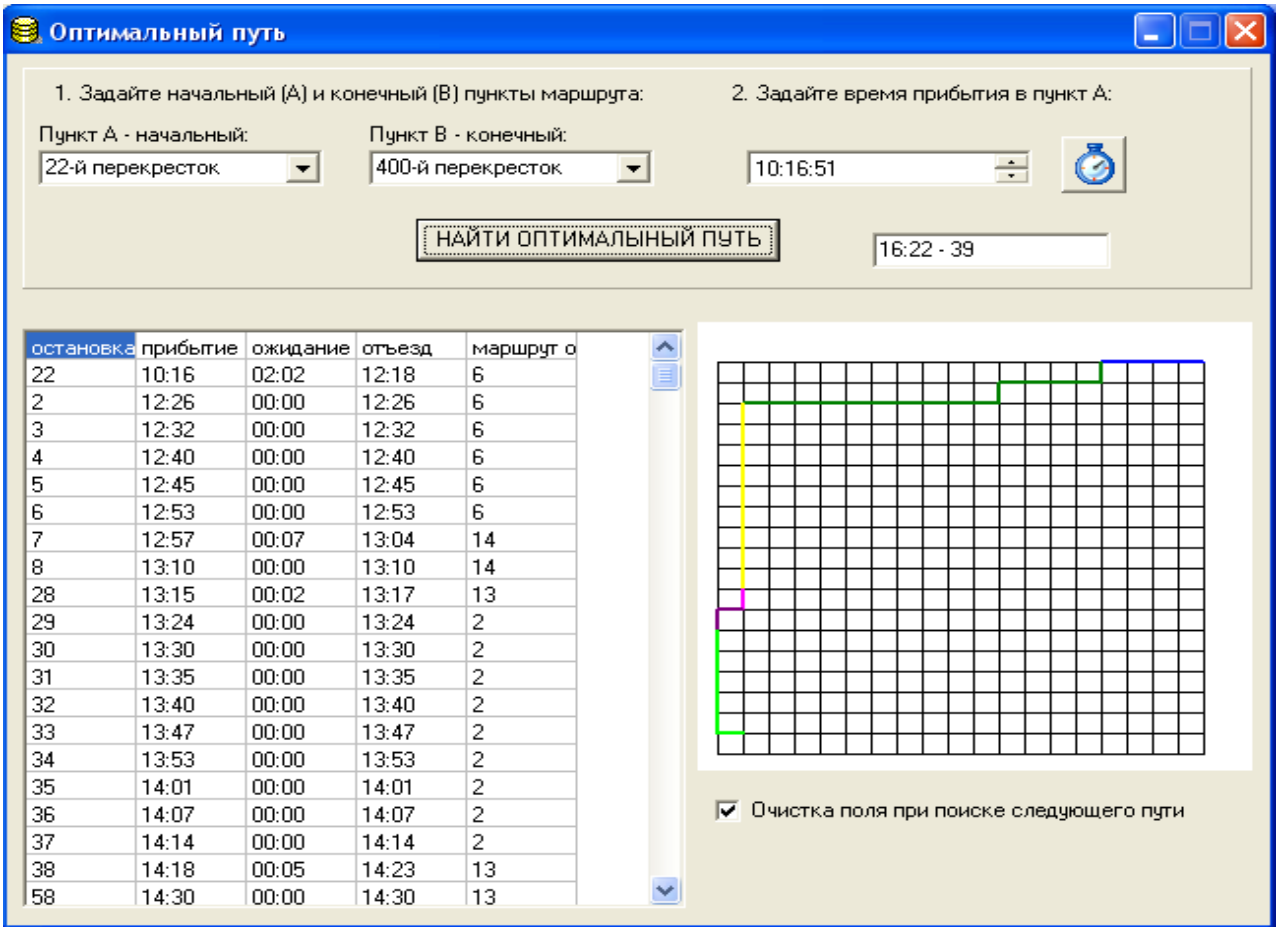
Первая рекурсивная функция *optima_time* – осуществляет поиск минимального времени, за которое можно добраться из пункта *A* в пункт *B*, точнее раннее время прибытия в пункт *B* при условии, что пассажир находится в пункте *A* в известный момент времени. Функция последовательно находит все остановки на расстоянии 1 перегона, в которые можно добраться из *A* и раннее время прибытия в них. Меткой для остановки является раннее время прибытия в нее. Если время прибытия найденной остановки меньше ее метки, то метка перезаписывается Далее функция вызывает сама себя для тех остановок, метки которых были перезаписаны, чтобы определить ранние времена прибытия в остановки, расположенные на расстоянии двух перегонов от *A*. И так далее, пока не достигнут критерий выхода из функции. Предусмотрено два критерия выхода:

1. Если не осталось ни одной остановки, для которой можно вызвать функцию *optima_way*, то есть при просмотре всех остановок очередного радиуса, ни одна из меток не изменилась.

2. Если метки всех просматриваемых остановок больше, чем метка точки *B*.

Вторая рекурсивная функция *optima_way* осуществляет так называемый обратный ход – по известным меткам остановок находит путь, следуя которому будет затрачено минимально возможное время. При первом вызове функция в качестве параметров получает остановку *B* и её метку. Функция обнаруживает все остановки, находящиеся на расстоянии одного перегона от *B*, из которых можно приехать в пункт *B* во время, равное содержащемуся в метке *B*, при этом время отъезда с остановки не должно быть больше метки этой остановки. Получив таким образом предпоследнюю остановку искомого пути, функция вызывает сама себя уже для этой остановки и её метки. И так далее, пока не будет достигнут критерий выхода – найденная остановка является начальным пунктом *A*.

Итоговая экранная форма представлена на рис. 3.15. Здесь перекрестки и соответствующие остановки пронумерованы снизу вверх и слева направо, начиная с 1 до 400. Оптимальная траектория изображается цветной ломаной, в которой цвет указывает на некоторый маршрут.



остановка	прибытие	ожидание	отъезд	маршрут о
22	10:16	02:02	12:18	6
2	12:26	00:00	12:26	6
3	12:32	00:00	12:32	6
4	12:40	00:00	12:40	6
5	12:45	00:00	12:45	6
6	12:53	00:00	12:53	6
7	12:57	00:07	13:04	14
8	13:10	00:00	13:10	14
28	13:15	00:02	13:17	13
29	13:24	00:00	13:24	2
30	13:30	00:00	13:30	2
31	13:35	00:00	13:35	2
32	13:40	00:00	13:40	2
33	13:47	00:00	13:47	2
34	13:53	00:00	13:53	2
35	14:01	00:00	14:01	2
36	14:07	00:00	14:07	2
37	14:14	00:00	14:14	2
38	14:18	00:05	14:23	13
58	14:30	00:00	14:30	13

Рис. 3.15. Результат поиска оптимального пути

Результаты моделирования показали, что данный алгоритм может быть перенесен на реальную транспортную сеть города Екатеринбурга с использованием актуальной информации о расписании движения.

3.12. АНАЛИЗ ЧУВСТВИТЕЛЬНОСТИ ОПТИМАЛЬНОГО ПУТИ К ВОЗМУЩЕНИЮ ИСХОДНЫХ ДАННЫХ

Как правило, пассажиры городского транспорта не имеют возможности влиять на составление расписания или маршрутов, а наоборот вынуждены подстраиваться под них. В таких условиях для более удобного использования ГПТ пассажиру было бы полезно знать, как изменится оптимальный путь и минимальное время для его преодоления при изменении начальных условий поездки. Например, может ли пассажир добраться до места назначения быстрее, если перейдет на соседнюю остановку с большим количеством маршрутов, выйдет из дома на несколько минут раньше или поедет не прямо до пункта назначения, а до соседней к нему остановки, а остаток пути преодолеет пешком.

Для учета подобных ситуаций проводится анализ чувствительности, то есть анализ того, как возможные изменения параметров исходной модели повлияют на полученное ранее оптимальное решение. Анализ чувствительности позволяет определить силу реакции результативного фактора на изменение зависимых факторов.

Здесь под анализом чувствительности понимается процедура, позволяющая определить, как и насколько изменятся оптимальный путь и его минимальное время при изменении одного из начальных параметров поиска:

- пункта отправления;
- времени отправления из начальной точки;
- пункта назначения.

Ниже представлены результаты анализа для каждого из параметров.

3.12.1. Изменение пункта отправления

Анализ чувствительности оптимального пути к возмущению положения остановки отправления показывает, насколько изменится оптимальный маршрут, если пассажир перейдет на другие близлежащие станции.

Пользователь задает расстояние, на которое может отклоняться начальный пункт A от исходного значения начального пункта A_0. В качестве результата анализа выдаются следующие сведения:

- сравнительная таблица оптимальных путей из возможных пунктов старта, ограниченных заданным расстоянием. Исходный путь отмечен желтым цветом;
- диаграмма, отражающая основные характеристики путей: время и длину в остановках;
- рисунок с полученными оптимальными путями. Насыщенность цвета отрезка показывает количество проходящих через него маршрутов;
- характеристики чувствительности оптимального времени в пути – максимальные увеличение и уменьшение оптимального времени на единицу изменения расстояния от A до A_0.

На рис. 3.16 показан анализ изменения оптимального пути при изменении начальной точки старта. Параметры анализа:

- время старта T = 12:43;
- пункт назначения B: «400-й перекресток»;
- исходный начальный пункт A_0: «86-й перекресток»;
- радиус отклонения начальной остановки A от исходной начальной остановки A_0: $r = \max |A - A_0| = 2$.

В качестве результата представлено сравнение оптимальных путей из 13 возможных точек старта A, которые расположены от исходной точки A_0 не дальше, чем на расстоянии двух остановок.

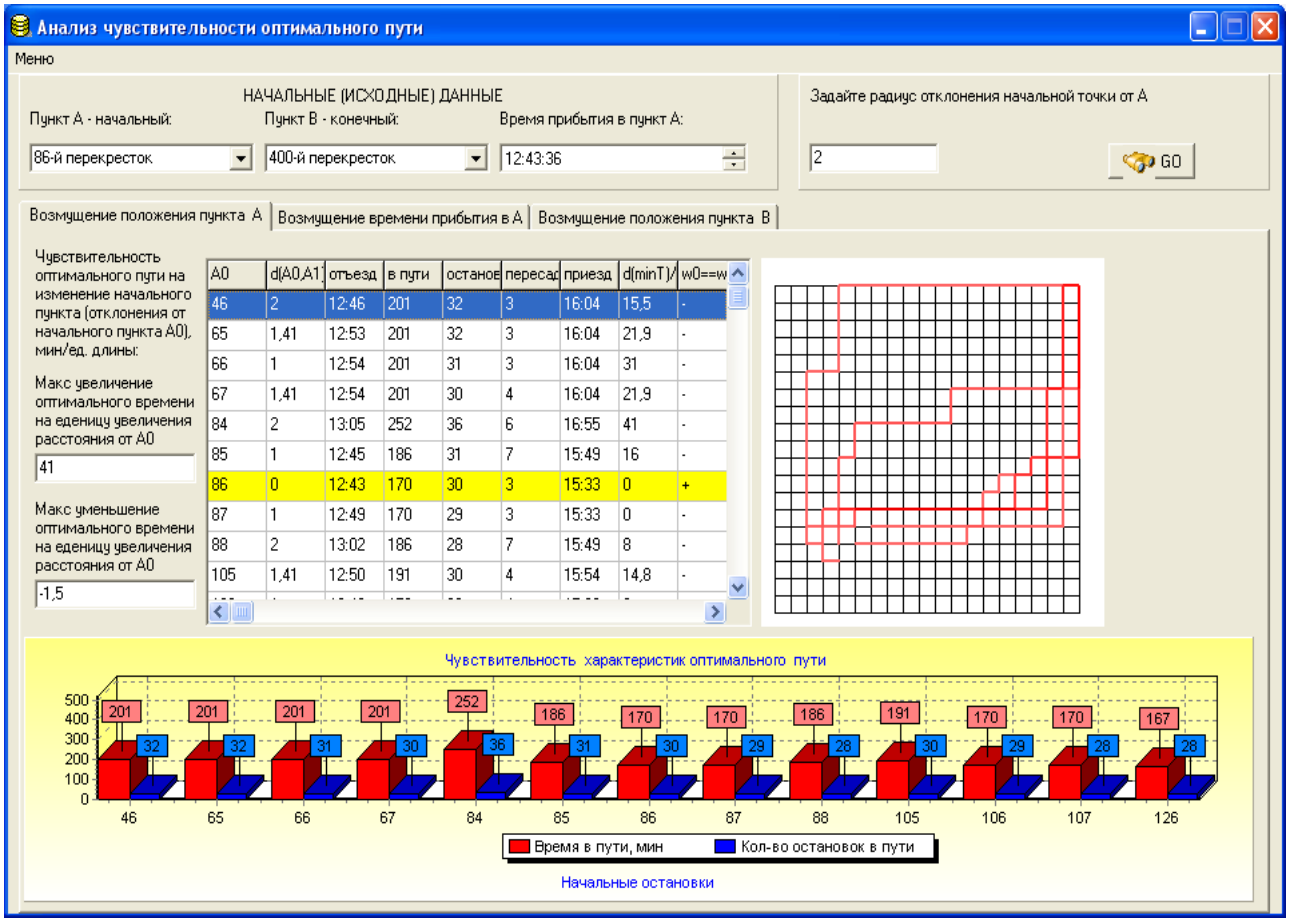


Рис. 3.16. Анализ зависимости оптимального пути от отклонения начальной остановки A

Пользователю доступен режим детального сравнения двух путей.

Пользователь выбирает из сравнительной таблицы интересующие пути. Первый выбранный путь получает красный цвет, второй – синий. Совпадающие отрезки в этих путях приобретают зеленый цвет. Результат сравнения представляется в таблице в качестве последовательности остановок, а на рисунке – в качестве траекторий с соответствующими цветами.

На рис. 3.17 показан результат использования режима детального сравнения двух оптимальных путей до остановки «400-й перекресток»:

1) из остановки «86-й перекресток» (красный цвет);

2) из остановки «105-й перекресток» (синий цвет).

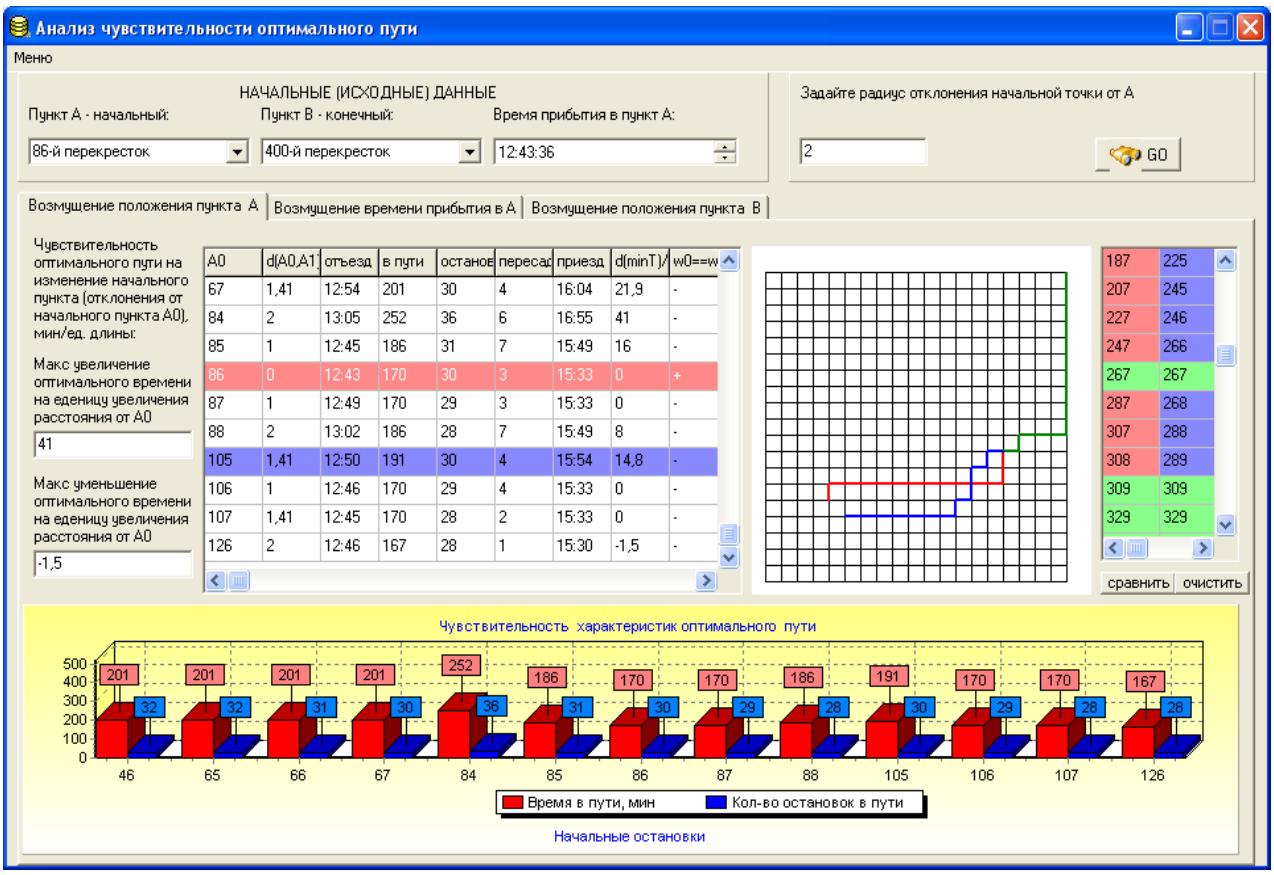


Рис. 3.17. Детальное сравнение двух путей

3.12.2. Изменение времени прибытия в пункт отправления

Анализ чувствительности оптимального пути к возмущению времени прибытия в пункт отправления (времени старта) показывает, насколько изменится оптимальный маршрут, если пассажир прибудет на остановку старта раньше или позже.

Пользователь задает временной интервал t, на который может отклоняться время старта T от исходного времени старта T_0 и шаг дискретизации этого интервала ΔT. Эти параметры характеризуют набор анализируемых времен старта $\{T_i\}$, как равномерно отстоящие моменты из интервала $[T_0 - t, T_0 + t]$.

В качестве результата анализа выдаются следующие сведения:

- сравнительная таблица оптимальных путей для возможных времен старта $\{T_i\}$. Исходный путь отмечен желтым цветом;
- график зависимости времени в пути от времени старта;
- график зависимости времени приезда в пункт назначения B от времени старта;
- график зависимости длины пути в остановках от времени старта;
- рисунок, на котором изображены все полученные оптимальные пути. Насыщенность цвета отрезка показывает количество проходящих через него маршрутов;

- характеристики чувствительности оптимального времени в пути – максимальные увеличение и уменьшение оптимального времени на единицу изменения времени старта.

На рис. 3.18 показан анализ изменения оптимального пути при изменении времени старта из начальной точки *A*.

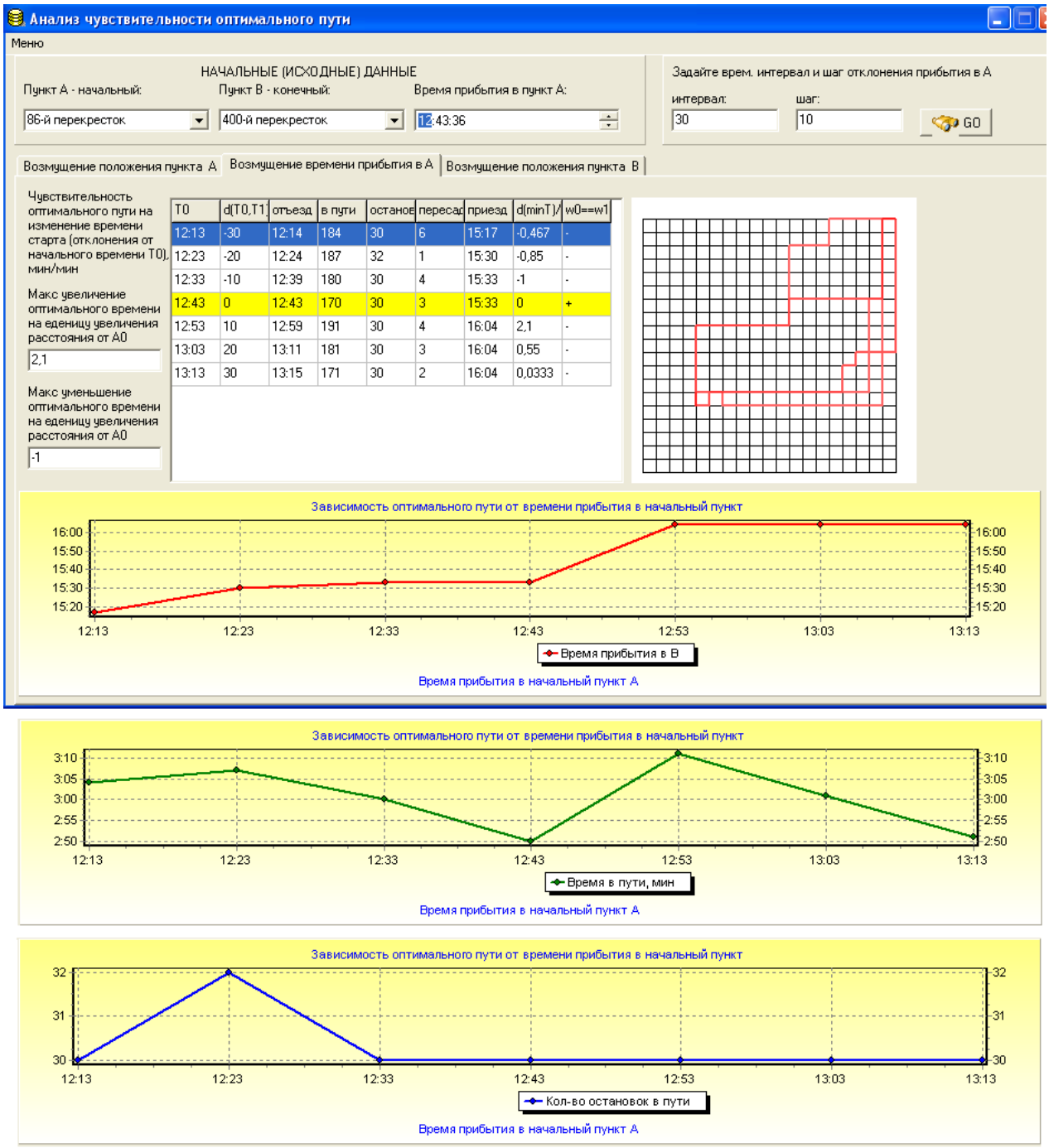


Рис. 3.18. Анализ зависимости оптимального пути от изменения времени старта

Параметры анализа:

- пункт отправления *A*: «86-й перекресток»;
- пункт назначения *B*: «400-й перекресток»;

- исходное время старта T_0 = 12:43;
- интервал, на который может отклоняться время старта T_i от исходного времени старта T_0: 30 мин., $|T_i - T_0| \le 30$ мин;
- шаг дискретизации этого интервала: ΔT = 10 мин.

В качестве результата представлено сравнение оптимальных путей для 7 возможных времен старта T_i, i = [1,7], которые отклоняются от исходного времени старта T_0 = 12:43 не более, чем на 30 мин.

Как и в первом случае, пользователю доступен режим детального сравнения двух путей.

3.12.3. Изменение пункта прибытия

Анализ чувствительности оптимального пути к возмущению положения остановки прибытия показывает, насколько изменится оптимальный маршрут, если пассажир поедет на транспорте не до самого пункта назначения, а до близлежащей к нему остановки, а остаток пути пройдет пешком.

Пользователь задает расстояние, на которое может отклоняться конечный пункт B от исходного значения конечного пункта B_0.

В качестве результата анализа выдаются следующие сведения:

- сравнительная таблица оптимальных путей для возможных пунктов прибытия, ограниченных заданным расстоянием. Исходный путь отмечен желтым цветом;
- диаграмма, отражающая основные характеристики путей: время и длину в остановках;
- рисунок с полученными оптимальными путями. Насыщенность цвета отрезка показывает количество проходящих через него маршрутов;
- характеристики чувствительности оптимального времени в пути: максимальные увеличение и уменьшение оптимального времени на единицу изменения расстояния от B до B_0.

На рис. 3.19 показан анализ изменения оптимального пути при отклонении конечной остановки B. Параметры анализа:

- время старта T = 13:42;
- пункт отправления A: «1-й перекресток»;
- исходный конечный пункт B_0: «317-й перекресток»;
- радиус отклонения конечной остановки B от исходной конечной остановки B_0: $r = \max(BB_0) = 2$.

В качестве результата представлено сравнение оптимальных путей в 11 возможных точек B, которые расположены от исходной точки B_0 не дальше, чем на расстоянии двух остановок.

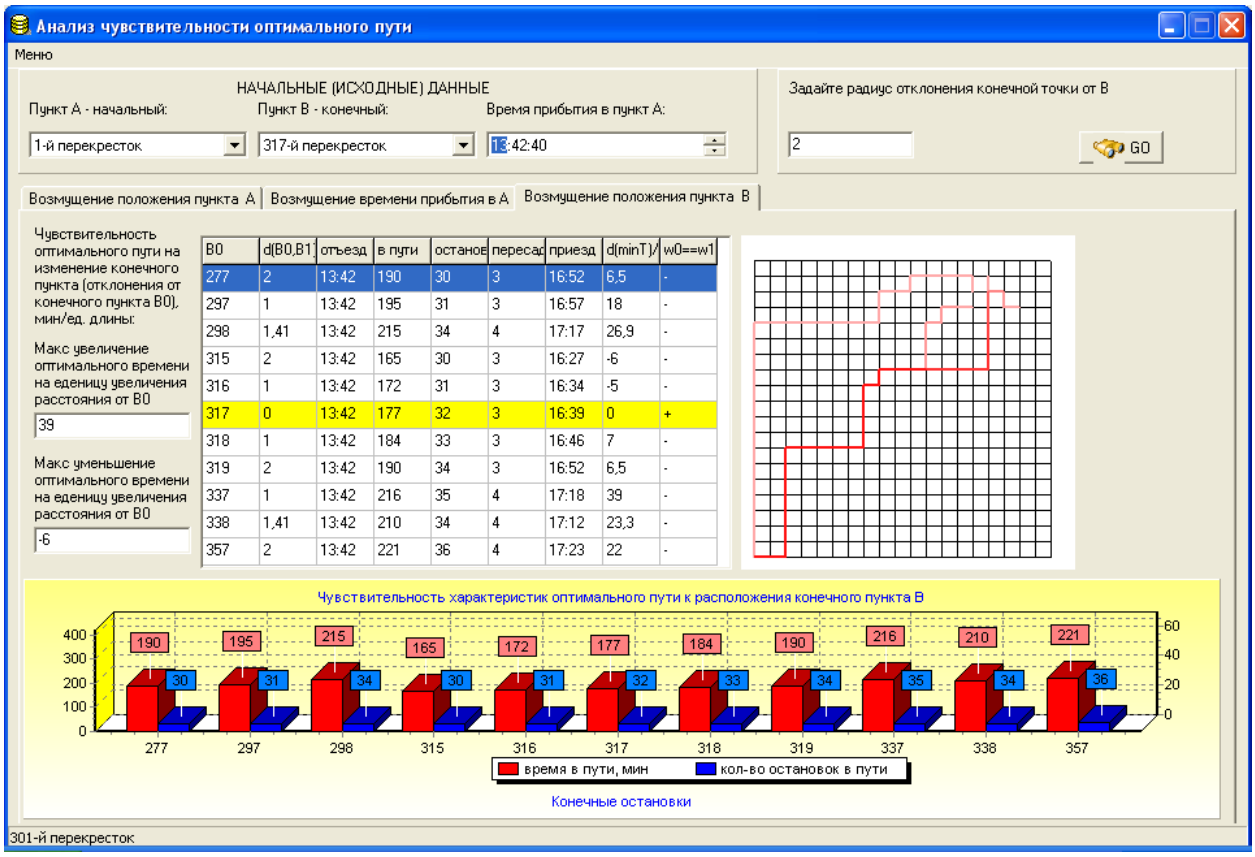


Рис. 3.19. Анализ зависимости оптимального пути от отклонения конечной остановки *B*

3.13. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ МОДЕЛИ

3.13.1. Исследование загруженности транспорта

Загруженность маршрута является его важной характеристикой и зависит от того, через какие районы города он проходит, какие социально-важные объекты лежат на его пути. Кроме того, загруженность маршрута меняется в течение суток.

Для пассажира знание загруженности того или иного маршрута полезно для того, чтобы по возможности избегать поездок в переполненном транспорте. Для транспортных предприятий загруженность маршрутов важна для более эффективного составления расписания, чтобы более полно обеспечивать спрос пассажиров на транспортные услуги.

Для вычисления загруженности маршрутов необходимо сопоставить расписания маршрутов («транспортного предложения») и такие данные, как цели, число поездок («транспортный спрос»). Расчет транспортного спроса – это самостоятельная задача, требующая трудоемких исследований инфраструктуры города, плотности населения районов и прочее. Эта задача в данном пособии не рассматривается, поэтому мы считаем, что спрос пассажиров на перевозки является известной величиной и задается количеством

запросов оптимального пути из одной точки в другую. То есть каждый запрос в системе оценивается как реальный спрос на данную поездку.

Рассмотрим расчет загруженности маршрутов на примере простой транспортной сети, показанной на рис. 3.20.

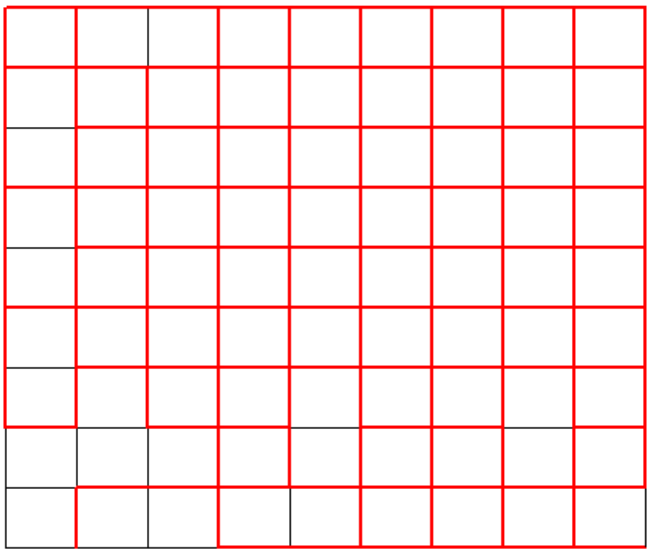


Рис. 3.20. Маршрутная карта транспортной сети

Наложим на маршрутную карту города произвольную карту инфраструктуры, точнее ее отдельную малую часть (рис. 3.21).

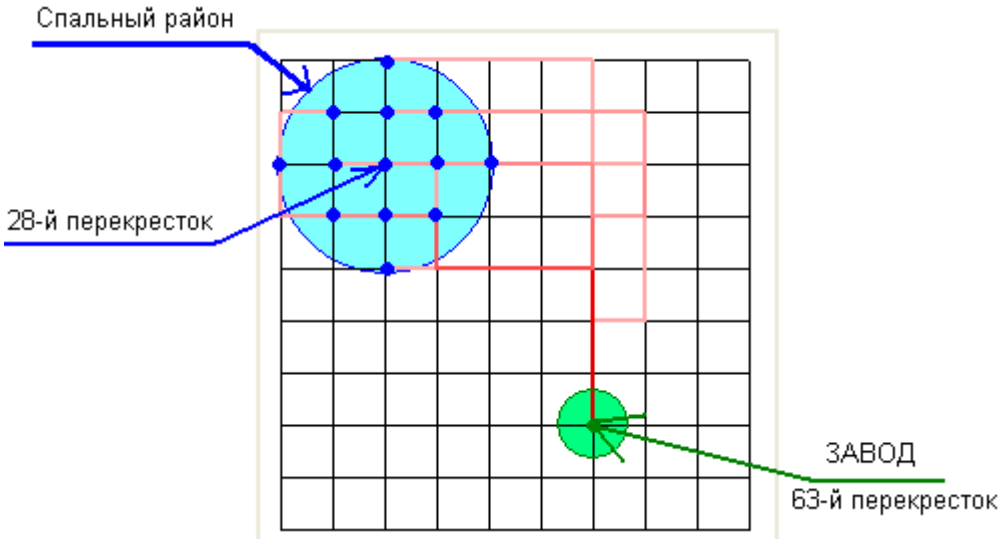


Рис. 3.21. Карта абстрактного года

Предположим, что 28-й перекресток является центром спального района, где находятся в основном жилые дома. Этот район на рис. 3.21 показан синим кругом. Кроме остановки «28-й перекресток», в районе имеется еще 11 остановок общественного транспорта.

Предположим, что большинство жителей района работают на городском заводе, который расположен на остановке «63-й перекресток». На рис. 3.21 заводская остановка показана зеленым цветом.

Каждое утро к 8 часам утра жители района едут на работу на этот завод.

На основе выше описанных данных смоделируем транспортный спрос жителей района, то есть их запросы на поиск оптимального пути. Запросы на оптимальный путь будут содержать следующие условия:

- остановка отправления – множество остановок района. Как видно из рис. 3.21, это все остановки, находящие от остановки «28-й перекресток» на расстоянии не более двух перекрестков;
- остановка назначения – «63-й перекресток»;
- время выхода на остановку (время старта) – 07:00.

Допустим, что район заселен равномерно и на каждую остановку района приходится равное количество рабочих завода, то есть равное количество запросов на оптимальный путь. Поэтому в модели можно произвести по одному запросу для каждой начальной остановки. В этом случае полученные результаты будут прямо пропорциональны реальным значениям загруженности. Для нахождения реальных показателей достаточно будет умножить найденные на число рабочих, приходящихся на одну начальную остановку.

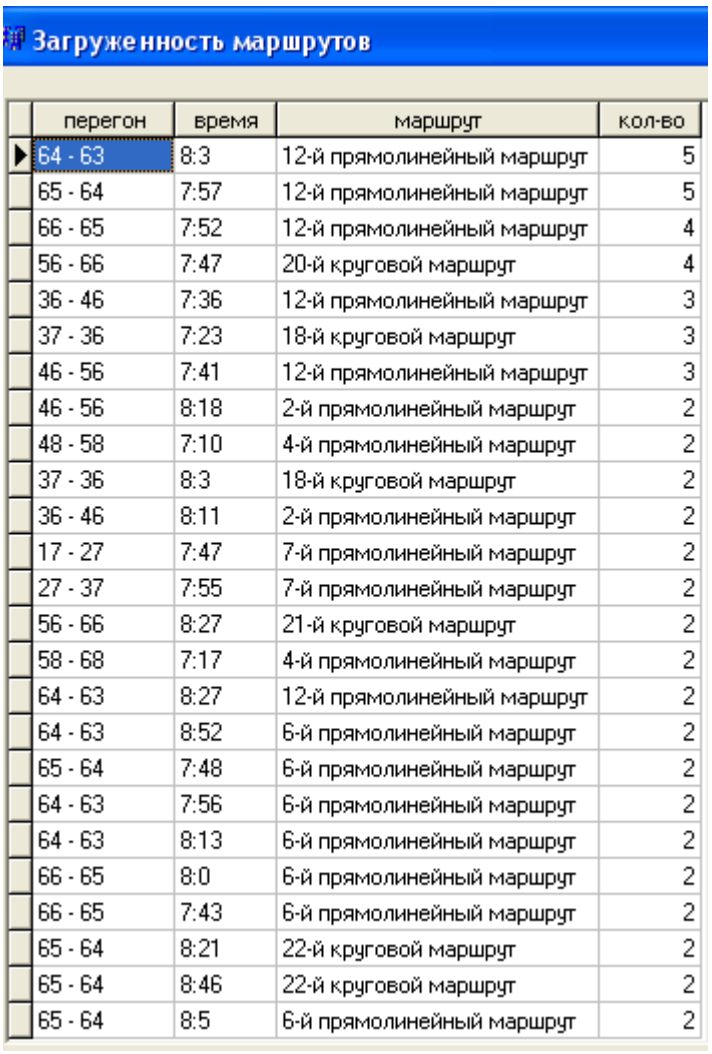
По заданным условиям организуем запрос, результатом которого является множество оптимальных путей: по одному пути из каждой остановки района – до заводской остановки. Результат запроса приведен на рис. 3.21 в виде траекторий маршрутов.

В модели реализована функция анализа загруженности маршрутов. Ее суть состоит в том, что каждому запросу сопоставляется пассажир, который едет именно по тому маршруту, который для него рекомендовала система. Таким образом, по всем имеющимся запросам ведется подсчет пассажиров, приходящихся на каждый конкретный состав общественного транспорта.

На рис. 3.22 представлен результат расчета загруженности маршрутов для рассматриваемого примера (по одному запросу на каждую начальную остановку).

На рис. 3.22 приведена утренняя статистика загруженности транспорта, перевозящего жителей спального района на завод. Показано количество пассажиров (колонка «кол-во»), находящихся в определенном составе общественного транспорта (колонка «маршрут») на момент его прохождения по конкретному перегону (колонка «перегон») в конкретное время (колонка «время»). Еще раз отметим, что пока результаты приведены при условии, что с каждой остановки района уезжает по одному рабочему.

В табл. 3.2 найденная информация представлена в более наглядном, обработанном виде для десяти наиболее загруженных составов. Здесь учтены все рабочие района: считаем, что в 07:00 с каждой остановки на завод уезжает по 15 рабочих.



перегон	время	маршрут	кол-во
64 - 63	8:3	12-й прямолинейный маршрут	5
65 - 64	7:57	12-й прямолинейный маршрут	5
66 - 65	7:52	12-й прямолинейный маршрут	4
56 - 66	7:47	20-й круговой маршрут	4
36 - 46	7:36	12-й прямолинейный маршрут	3
37 - 36	7:23	18-й круговой маршрут	3
46 - 56	7:41	12-й прямолинейный маршрут	3
46 - 56	8:18	2-й прямолинейный маршрут	2
48 - 58	7:10	4-й прямолинейный маршрут	2
37 - 36	8:3	18-й круговой маршрут	2
36 - 46	8:11	2-й прямолинейный маршрут	2
17 - 27	7:47	7-й прямолинейный маршрут	2
27 - 37	7:55	7-й прямолинейный маршрут	2
56 - 66	8:27	21-й круговой маршрут	2
58 - 68	7:17	4-й прямолинейный маршрут	2
64 - 63	8:27	12-й прямолинейный маршрут	2
64 - 63	8:52	6-й прямолинейный маршрут	2
65 - 64	7:48	6-й прямолинейный маршрут	2
64 - 63	7:56	6-й прямолинейный маршрут	2
64 - 63	8:13	6-й прямолинейный маршрут	2
66 - 65	8:0	6-й прямолинейный маршрут	2
66 - 65	7:43	6-й прямолинейный маршрут	2
65 - 64	8:21	22-й круговой маршрут	2
65 - 64	8:46	22-й круговой маршрут	2
65 - 64	8:5	6-й прямолинейный маршрут	2

Рис. 3.22. Загруженность маршрутов

Таблица 3.2

Результаты 10 наиболее загруженных транспортных составов

Маршрут	Время	Перегон	Количество пассажиров транспортного состава, в модели	Реальное количество пассажиров в транспортном составе
12	08:03	64 – 63	5	75
12	07:57	65 – 64	5	75
12	07:52	66 – 65	4	60
20	07:47	56 – 66	4	60
12	07:36	36 – 46	3	45
18	07:23	37 – 36	3	45
12	07:41	46 – 56	3	45
2	08:18	46 – 56	2	30
4	07:10	48 – 58	2	30
18	08:03	37 – 36	2	30

Информация табл. 3.2. может быть интерпретирована следующим образом. В утреннее время наиболее загруженным является 12-й маршрут. Наибольшее количество пассажиров, одновременно находящихся в транспортном составе 12-го маршрута, составляет 75 человек и приходится на то время, когда состав проезжает перегоны между 65-м и 64-м перекрестками (07:57) и 64-м и 63-м (08:03) перекрестками. Когда 12-й маршрут проезжает перегоны между 46-м и 56-м перекрестками (07:41) и 36-м и 46-м перекрестками (07:36), в салоне находится 45 пассажиров.

Приведенный пример является очень упрощенным и содержит множество допущений, которые в реальной жизни не всегда выполняются. Но при наличии достоверной информации о величине транспортного спроса на все маршруты, рассчитанная в системе загруженность транспортных средств будет соответствовать действительной ситуации.

3.13.2. Оценка эффективности транспортной сети

На сегодняшний день в нашей стране выделяют четыре основных составляющих, по которым оценивается работа транспортного предприятия и обслуживаемой им транспортной сети:

- вагонокилометры;
- регулярность движения;
- выручка;
- выпуск в час пик.

Все вышеперечисленные критерии рассчитываются для реальных транспортных сетей по факту их работы. В разрабатываемой модели не ведется учет пассажиров и контроль фактического соблюдения планов работы общественного транспорта. Поэтому указанные характеристики не могут быть применены для оценки моделируемой транспортной сети.

Для объективной оценки транспортной сети, смоделированной в разработанной системе, предложены следующие критерии:

1. «Покрытие города» – процентное отношение количества перекрестков, по которым существует движение ГПТ, к количеству всех перекрестков города. Эта величина показывает обеспеченность города транспортными связями.

2. «Гарантированное время поездки» – минимальное время, за которое пассажир гарантированно может добраться из любой точки города в любую другую точку. При этом точкой считается любой перекресток города, по которому существует движение общественного транспорта. Данное время гарантировано, если пассажир для выбора маршрута использует рекомендации системы. Значение этого критерия рассчитывается для трехдневных периодов: утро (06:00 – 11:00), день (11:00 – 17:00), вечер (17:00 – 21:00). Эта величина характеризует эффективность составленного расписания, регулярность движения.

3. «Количество пересадок» – минимальное количество пересадок с одного транспортного маршрута на другой, совершив которые пассажир гарантированно может добраться из любой точки города в любую другую точку. Эта величина характеризует, насколько удобно спланированы маршруты по отношению к городской инфраструктуре.

4. «Время ожидания» – усредненная величина ожидания пассажира транспорта на остановке за одну поездку из любой точки города в любую другую точку. При этом учитывается ожидание пассажира не только на остановке отправления, но и на всех остановках, где осуществляется пересадка. Эта величина характеризует эффективность составленного расписания, регулярность движения.

На рис 3.23 показаны результаты расчета предложенных критериев для транспортной сети, представленной на рис. 3.20.

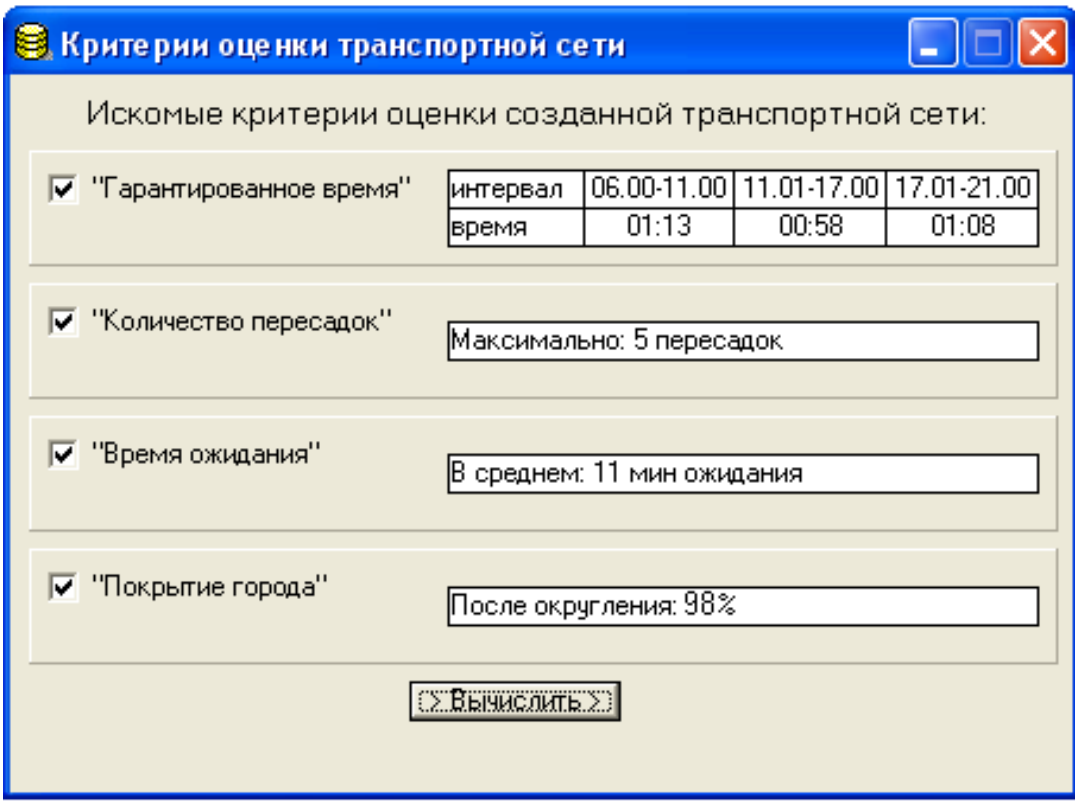


Рис. 3.23. Расчет критериев оценки транспортной сети

Результаты оценки транспортной сети можно трактовать следующим образом. 98 % перекрестков города оборудованы остановками общественного транспорта, через которые существует постоянное движение. Действительно, на рис. 3.20 показано, что транспортное движение отсутствует только по двум перекресткам из ста, что составляет 2 %. Чтобы добраться из любой точки города в любую другую, пассажиру понадобится совершить не более пяти пересадок. При этом в среднем за поездку ему придется 11 минут ждать транспорт на остановках. В утреннее время пассажир затратит на поездку не более, чем 1 час 13 минут, днем – не более, чем 58 минут, вечером – не более, чем 1 час 8 минут.

3.13.3. Учет дорожной ситуации

В п. 3.7 подробно освещен вопрос учета дорожной ситуации при информировании населения о работе общественного транспорта.

В разработанной авторской модели учет дорожной ситуации реализован следующим образом.

1. Программный модуль *Closing_roads* (Закрытие дороги) предоставляет пользователям модели два вида доступа к информации о дорожной ситуации:

- защищенный паролем доступ к перекрытию и открытию движения по перекресткам города показан на рис. 3.24;

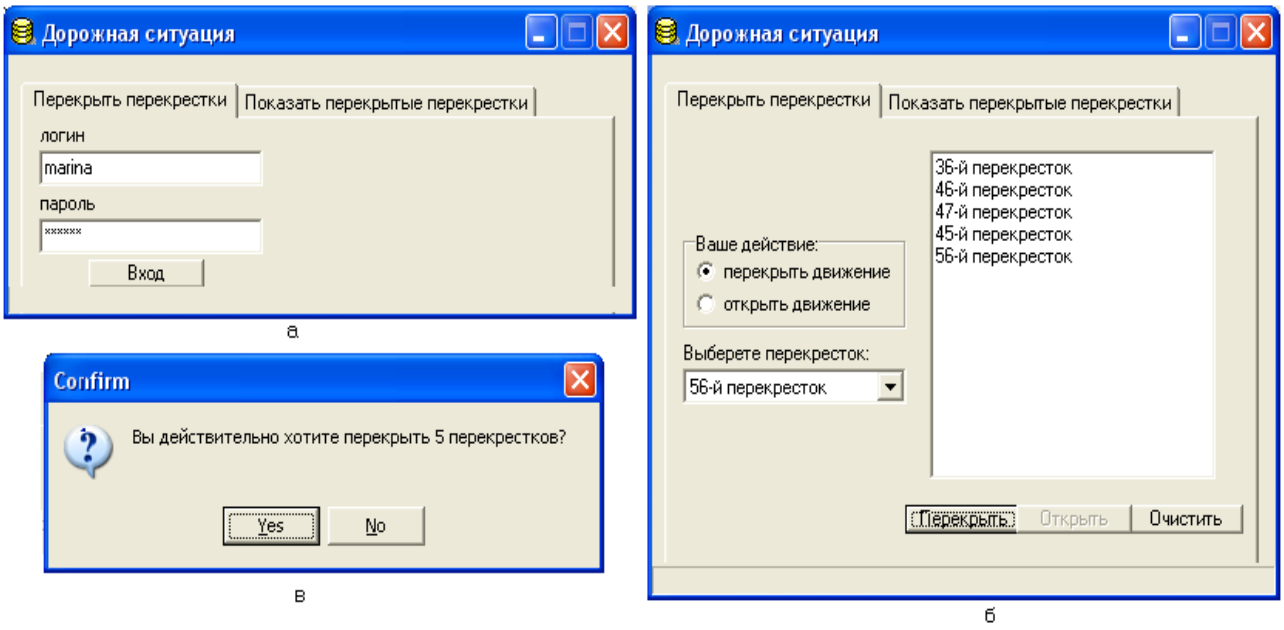


Рис. 3.24. Процесс перекрытия движения:
а) запрос пароля на доступ к изменениям дорожной ситуации;
б) выбор перекрываемых перекрестков;
в) запрос подтверждения

- свободный доступ для всех пользователей к просмотру карты транспортной сети с указанием тех участков дороги, которые на текущий момент являются закрытыми для движения общественного транспорта, показан на рис. 3.25. Закрытые перекрестки показываются на карте красным перечеркнутым кругом.

2. В программном модуле *Optimum_way* (Оптимальный путь) пользователям предоставляется возможность найти оптимальный путь между двумя точками города с учетом дорожной ситуации (перекрытых перекрестков).

Из рис 3.26 видно, что при учете дорожной ситуации результат запроса оптимального пути изменяется. Новый маршрут строится так, чтобы в нем не участвовали закрытые перекрестки.

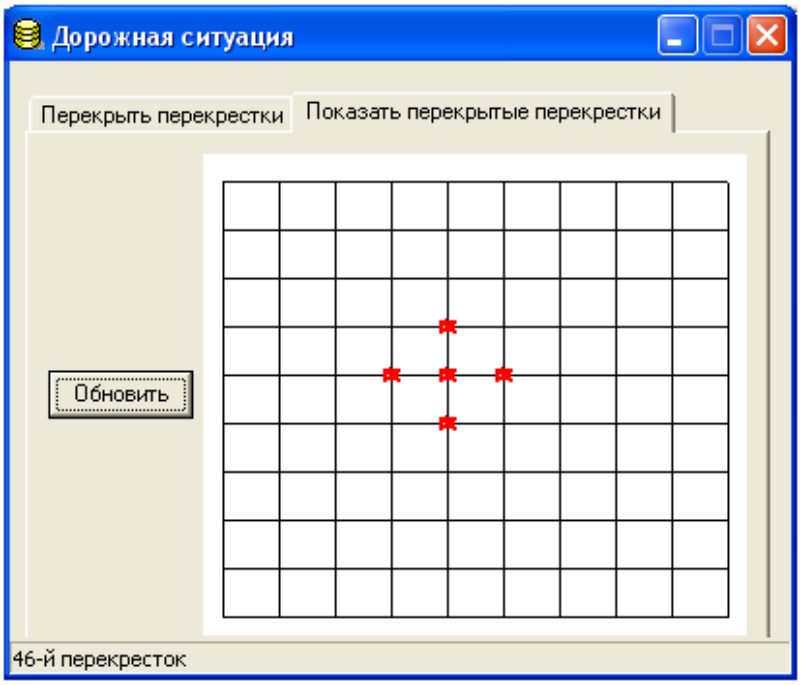


Рис. 3.25. Просмотр закрытых участков дороги

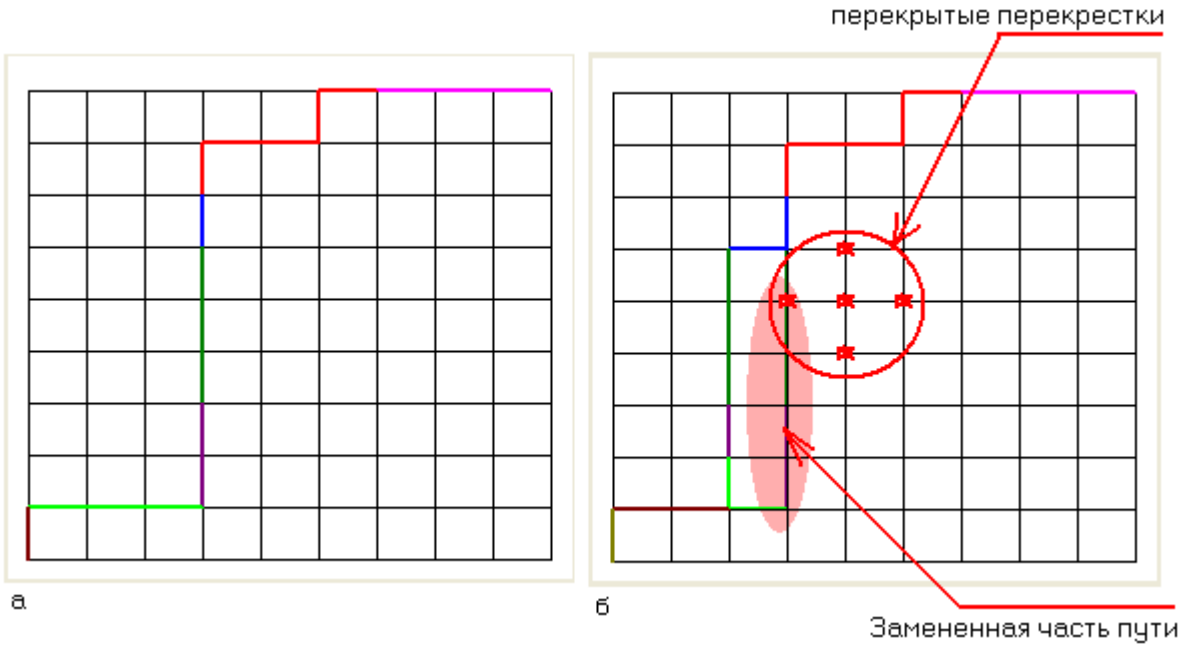


Рис. 3.26. Поиск оптимального пути с учетом и без учета дорожной ситуации:
а) дорожная ситуация не учитывается; б) дорожная ситуация учитывается

3.14. ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ ДВИЖЕНИЕМ ТРАНСПОРТА НА ПЕРЕКРЕСТКАХ

В данном параграфе рассматривается авторский новый способ регулирования движения автотранспорта на перекрестке, как интеллектуальная система управления движением транспорта на перекрестках.

Наиболее часто используемые на практике способы регулирования движения автотранспорта на перекрестке [3.20 – 3.22] работают следующим образом. Переключение сигналов светофора проводится с использованием

переключающего реле и таймера с фиксированной установкой времени переключения сигналов светофора. Время переключения может корректироваться по месту расположения светофора или дистанционно.

Однако недостатком этого способа является установка фиксированного времени переключения, что не позволяет в динамике (в темпе с обстановкой трафика) отслеживать складывающуюся ситуацию движения и своевременно реагировать на эту ситуацию, используя длительность времени переключения сигналов светофора. В свою очередь это создает несимметричные потоки автомашин по сторонам движения, что приводит к неоправданным задержкам движения и возникновению пробок на пути движения автотранспорта.

При более сложном графике регулирования потоков транспорта на перекрестках используются детекторы обнаружения транспортных сред и учитывается длина очереди автомашин перед светофором [3.20 – 3.22]. Существуют также интеллектуальные системы моделирования управления транспортными потоками на перекрестках [3.23, 3.24].

Однако недостатком этих способов и устройств является в одном случае отсутствие четких рекомендаций для определения времени переключения светофора, а в другом – сложности предлагаемых средств распределения потоков транспорта, что в итоге не позволяет организовать работу переключающих устройств управления светофора в реальном времени. В случаях [3.23, 3.24], например, алгоритмы и программное обеспечение предназначены в основном для моделирующих устройств.

В известном способе и устройстве автоматизированным управлением уличным транспортом [3.21] учитывается длина участка дороги, занятой автомашинами перед перекрестком. Однако при этом при скоплении машин перед перекрестком не учитывается расстояние между машинами на перекрестке, т. е. число машин на данном участке дороги, и тем самым не учитывается время задержки на перекрестке движения последующей машины после начала движения предыдущей, т. е. реальное время чистого (транспортного) запаздывания, что не позволяет реально оценивать требуемое время переключения сигналов светофора.

Технической задачей системы управления движением транспорта на перекрестках является обеспечение получения достоверной информации о времени запаздывания при прохождении автотранспорта на длине участка дороги, занятой автотранспортом для более точной оценки необходимого времени переключения сигналов светофора и тем самым более эффективного управления движения автотранспорта на перекрестке, снижение простоев транспорта, уменьшение расхода горючего и вредных выбросов в атмосферу.

Указанная задача достигается тем, что система управления движением транспорта на перекрестках включает регулирование движения с помощью светофора, переключение сигналов светофора с использованием переключающих реле и таймера, определение длины участка дороги, занятой автомашинами от ограничивающей линии перекрестка до конечной границы

участка дороги, занятой автомашинами на данном участке дороги, отличается тем, что время переключения сигналов светофора с зеленого света на красный устанавливают с учетом определения среднего расстояния между машинами перед перекрестком, т.е. числа машин на данном участке дороги и с учетом времени задержки на перекрестке движения последующей автомашины после начала движения предыдущей машины, при этом время переключения сигналов светофора с зеленого света на красный $t_{\text{пер}}$ определяется одновременно для встречных полос движения на перекрестке из выражения

$$t_{\text{пер}} = t_{\text{уст}} + \left[\frac{\Delta}{W} + (n-1)\tau_{\text{тр}} \right], \qquad (3.4)$$

где $t_{\text{уст}}$ – базовое время установки переключения сигналов светофора для обычного режима движения (без скопления автомашин перед светофором); Δ – длина участка дороги, занятого автомашинами; W – средняя скорость движения машин на перекрестке; n – число автомашин перед светофором; $\tau_{\text{тр}}$ – среднее время задержки начала движения последующей машины после начала движения предыдущей, после чего сравнивают время переключения сигналов светофора для противоположных сторон движения автомашин и устанавливают на светофоре наибольшее из этих сравниваемых значений $t_{\text{пер}} = \max t_{\text{пер}}$.

Система управления движением транспорта на перекрестках также отличается тем, что в случае наличия двух или нескольких полос движения перед перекрестком время переключения сигналов светофора выбирают наибольшим из этих для этих сравниваемых значений.

Известно, какие трудности при автоматическом регулировании возникают, если в регулируемом объекте имеется время чистого (транспортного) запаздывания. При этом необходимо иметь точную информацию об этом времени для синтеза системы регулирования и оценки времени регулирования процессов [3.24].

В интеллектуальной системе управления движением транспорта на перекрестках в сложной обстановке дорожного трафика (при скоплении автомашин перед перекрестком) для уточнения этого времени чистого запаздывания учитывается не только длина участка дороги Δ, занятая автомашинами, но и среднее расстояние между машинами и число машин перед перекрестком. Это объясняется тем, что время чистого запаздывания движения данного скопления автомашин перед перекрестком определяется не только скоростью движения машин после установления зеленого сигнала светофора, но и установившимся расстоянием между машинами, а также временем определенной неизбежной задержки движения (трогания) последующей машины после начала движения впереди стоящей машины (время начала трогания машины) $\tau_{\text{тр}}$. Среднее расстояние между машинами на длине Δ при этом определится числом машин n, а дополнительное время при задержке движения $t_{\text{доп}}$ определится как

$$t_{доп} = (n - 1)\ \tau_{тр}. \quad (3.5)$$

В этом случае общее время чистого запаздывания $t_{зап}$ нахождения данной очереди машин перед перекрестком определится как

$$t_{зап} = \frac{\Delta}{W} + (n-1)\tau_{тр}, \quad (3.6)$$

где Δ – длина дороги, занятая автотранспортом (наибольшая из встречных полос движения); W – средняя скорость движения машин на перекрестке.

В интеллектуальной системе управления движением транспорта на перекрестках для организации бесперебойной работы регулирования движением на перекрестке введено время чистого запаздывания $t_{зап}$, как корректирующее время к определенному (базовому) времени $t_{уст}$ установки переключения сигнала светофора для обычного режима движения, т. е. времени, устанавливаемом без скопления автомашин перед светофором. В этом случае при наличии установки базового времени даже в случае непредвиденного сбоя в определении времени чистого запаздывания $t_{зап}$ регулирование движения не будет полностью блокировано. В данной системе величины $t_{уст}$, W, $t_{зап}$ являются корректируемыми уставками, так как их значения зависят от конкретной обстановки на данном участке дороги (например, гололед, мокрая дорога, неровное покрытие дороги и т. д.). Величины Δ и n при использовании датчиков (детекторов) наблюдения определяются с использованием методов распознавания образов.

В интеллектуальной системе управления движением транспорта на перекрестках требуемое время переключения $t_{пер}$ определяют одновременно для встречных полос движения на перекрестке и сравнивают эти определяемые времена с выбором для переключения сигналов на светофоре наибольшего из них, т. е. с выбором значения

$$t_{пер} = \max t_{пер}. \quad (3.7)$$

В случае наличия на перекрестке двух или нескольких полос движения выбирают время переключения сигналов светофора по наибольшему значению для этих полос движения.

В случае наличия на светофоре стрелок поворотов движения (вправо, влево) время переключения сигналов светофора определяют дополнительно для данной полосы движения автомашин.

В интеллектуальной системе управления движением транспорта на перекрестках время установки переключения сигналов светофора для обычного режима движения, средняя скорость движения машин на перекрестке, среднее время задержки начала движения последующей автомашины после начала движения предыдущей являются корректирующими величинами при определении времени переключения сигналов светофора, оцениваются и устанавливаются операторами в зависимости от условий обстановки движения перед перекрестком.

Таким образом, в случае скопления автомобилей перед перекрестком время переключения увеличивается на реальное время чистого запаздывания, что способствует быстрой эвакуации автомобилей на перегруженной трафиком стороне дороги перед перекрестком, предотвращает дальнейшее скопление автотранспорта и образование дорожных пробок.

Интеллектуальная система управления движением транспорта на перекрестках реализуется устройством, представленным на рис. 3.27 и 3.28. Оно включает (рис. 3.27) регулирующий светофор 1, датчик наблюдения за отрезками дороги перед светофорами в перпендикулярных (или требуемых) направлениях 2; блок светофора с датчиками (детекторами) наблюдения 3; линию передачи показаний датчиков наблюдения к обрабатывающему приемнику сигналов 4, пункт управления сигналами светофора 5; центральные оси визирования датчиков наблюдения 6; линии учета результатов визирования датчиков наблюдения 7; участки дороги, занятые автомобилями 8; устройство сканирования датчиков наблюдения 9.

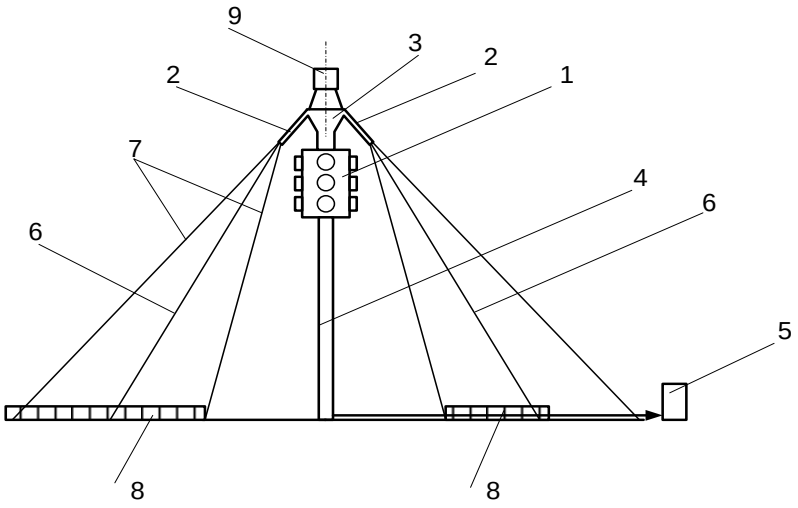


Рис. 3.27. Устройство, реализующее способ регулирования движения автотранспорта на перекрестке

На рис. 3.28 представлена блок-схема интеллектуальной системы управления движением транспорта на перекрестках, включающая светофор 1; датчик (детектор) наблюдения 2; блок светофора с датчиками наблюдения 3; линии передачи сигналов от датчиков наблюдения 4 к обрабатывающему приемнику сигналов 5; блок распознавания длины наибольшего участка дороги, занятой автотранспортом по данному направлению и числа автомашин на данном участке дороги 6; вычислительный блок 7; блок корректировки: базового времени переключения сигналов светофора (при отсутствии скопления автомашин перед перекрестком), средней скорости движения автотранспорта перед перекрестком и времени задержки начала движения

последующей автомашины после начала движения предыдущей 8; блок установки расчетного времени переключения сигналов светофора 9, таймер 10; реле переключений 11 и блок сканирования датчиков наблюдения 12.

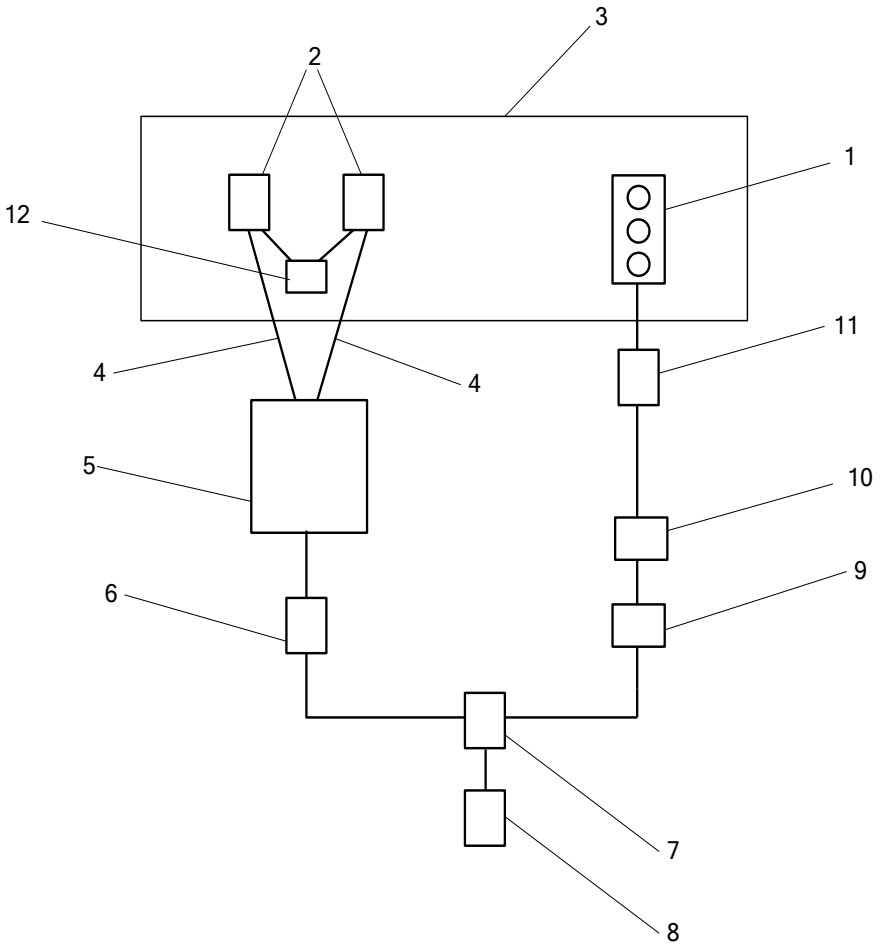


Рис. 3.28. Блок-схема интеллектуальной системы управления движением транспорта на перекрестках

Интеллектуальная система управления движением транспорта на перекрестках работает следующим образом. На светофоре 1 (рис. 3.27 и 3.28) с использованием таймера 10 (рис. 3.28) устанавливается базовое время переключения сигналов светофора $t_{уст}$ (см. формулу (3.4)), при этом реле 11 (см. рис. 3.28) обеспечивает переключение сигналов светофора. Датчики наблюдения 2 (см. рис. 3.27 и 3.28) обеспечивают визирование на участки дороги перед светофором и передачу изображения участка дороги с находящимися на ней автомобилями по линиям передачи 4 в обрабатывающий приемник сигналов 5 (см. рис. 3.28). С использованием блока 6 (см. рис. 3.28), в котором используется программа распознавания образов, определяется длина участка дороги 8 (см. рис. 3.27), занятого автотранспортом от ограничивающей линии перед перекрестком до конечной границы участка дороги, занятой автомашинами Δ, наибольшая для данного направления движения и число машин на данном участке дороги. В вычислительном блоке 7 (см. рис. 3.28)

определяется время переключения сигналов светофора от зеленого света к красному по формуле (3.4) для каждого из противоположных направлений движения, при этом коэффициенты $t_{уст}$, W, $\tau_{пр}$ задаются и могут при необходимости корректироваться блоком корректировки 8 (см. рис. 3.28). В вычислительном блоке также производится сравнение определяемых времен переключения $t_{пер}$ для противоположных направлений движения на перекрестке с выбором для использования наибольшего значения $t_{пер} = \max t_{пер}$. Вычисленное и выбранное в блоке 7 время переключения сигналов светофора $t_{пер} = \max t_{пер}$ (см. формулу (3.4)) используется в блоке 9 для корректировки задания таймеру 10, который переключает с помощью реле 11 сигналы светофора 1 в соответствии со скорректированным временем переключения. В предлагаемой интеллектуальной системе управления движением транспорта на перекрестках используются в качестве датчиков 3 (см. рис. 3.27 и 3.28) и обрабатывающего приемника сигналов 5 (см. рис. 3.28) различные виды их исполнения (по выбору изготовителя): визирные головки телевизионных установок и телевизионный приемник; датчики восприятия инфракрасного излучения и соответствующие приемники; излучатели радиоволн и радиоприемники. В случае потребности изменять центральную ось визирования датчиков 6 и соответствующие линии углов визирования датчиков 7 (см. рис. 3.27) используют сканирующее устройство 9 (см. рис. 3.27), 12 (см. рис. 3.28).

Рассмотрим примеры определения параметров интеллектуальной системы управления движением транспорта на перекрестках.

Пример 1. В блоке 8 (см. рис. 3.28) установлено базовое время переключения сигналов светофора $t_{уст} = 10$ с (см. формулу (3.4)).

С использованием датчиков наблюдения 2, линий передачи 4 и обрабатывающего устройства 5 определено расстояние от ограничивающей линии перед светофором до конечной границы участка дороги, занятой автомашинами на полосе движения одного направления дороги 8 (левая часть рис. 3.27) $\Delta = 30$ м и число машин $n = 12$, а для противоположного направления дороги $\Delta = 20$ м и $n = 10$. В блоке 8 (см. рис. 3.28) установлены параметры в формуле (3.4) $t_{уст} = 10$ с, $W = 20$ м/с и $\tau_{пр} = 2$ с. В блоке 7 (см. рис. 3.28) по формуле (3.4) вычислено требуемое время переключения сигналов светофора с зеленого света на красный для обоих направлений движения

$$t_{пер.1} = 10 + \left[\frac{30}{20} + (12-1)\cdot 2\right] = 33{,}5 \text{ с};$$

$$t_{пер.2} = 10 + \left[\frac{20}{20} + (10-1)\cdot 2\right] = 29 \text{ с}.$$

В блоке 7 происходит сравнение этих величин и выбирается наибольшее значение из них $t_{пер.1} > t_{пер.2}$

$$t_{пер.1} = \max t_{пер} = 33{,}5 \text{ с}.$$

С использованием блока установки расчетного времени переключения сигналов светофора 9 (см. рис. 3.28) время переключения сигналов светофора на этот полупериод переключения установлено в таймере 10 $t_{\text{пер}}$ = 33,5 с, которое и реализуется с помощью реле 11 (см. рис. 3.28). После переключения сигнала светофора в следующем полупериоде работы светофора последовательность рассмотренных выше действий устройства повторяется, но уже для полосы дороги, перпендикулярной полосе, рассмотренной в предыдущем полупериоде переключения.

Из этого примера следует, что длительность скорректированного времени переключения светофора зависит в значительной степени от времени задержки трогания машин после начала движения предыдущей машины, что и предусматривается в данной интеллектуальной системе управления движением транспорта на перекрестках.

Пример 2. Базовое время переключения то же, что и в примере 1 $t_{\text{уст}}$ = 10 с, а также те же значения Δ = 30 м и n = 12 и Δ = 20 м и n = 10 м. Из-за наличия гололеда и тумана скорость движения автотранспорта на перекрестке снижена до 10 м/с, а время задержки $\tau_{\text{пр}}$ увеличено до 2,5 с.

В этом случае

$$t_{\text{пер.1}} = 10 + \left[\frac{30}{10} + (12-1)\cdot 2{,}5\right] = 40{,}5 \text{ с};$$

$$t_{\text{пер.2}} = 10 + \left[\frac{20}{10} + (10-1)\cdot 2{,}5\right] = 34{,}5 \text{с}.$$

В этом случае время переключения сигналов светофора увеличивается по сравнению с примером 1 и равняется

$$t_{\text{пер}} = \max t_{\text{пер}} = 40{,}5 \text{ с}.$$

Пример 3. Базовое время переключений то же, что и в примере 1 $t_{\text{уст}}$ = 10 с. С использованием датчика наблюдения 2, линий передачи 4 и обрабатывающего устройства (см. рис. 3.27, 3.28) не установлено наличие транспорта на участках дороги от ограничивающей линии перед перекрестком по двум противоположным направлениям движения. В этом случае Δ = 0 и по формуле (3.4) определено в блоке 7 (см. рис. 3.28) время переключения $t_{\text{пер}}$, равное базовому времени переключения

$$t_{\text{пер.1}} = t_{\text{пер.2}} = t_{\text{пер}} = t_{\text{уст}} + 0 \cdot 0{,}3 = 10 \text{ с}.$$

Применение данной интеллектуальной системы управления движением транспорта на перекрестках обеспечит ускоренную эвакуацию транспорта с полосы дороги, наиболее перегруженной автотранспортом, что предотвратит образование больших скоплений автотранспорта и образование неоправданных транспортных пробок. Тем самым снизится расход горючего автомашин и эмиссия вредных выбросов на участках дорог перед перекрестками.

3.15. КОНТРОЛЬНЫЕ ВОПРОСЫ И УПРАЖНЕНИЯ

Вопрос 1. Каковы основные отличия инструментальной среды от информационной системы? Приведите примеры.

Вопрос 2. Какие критерии качества сети муниципального транспорта требуют наличие временно*г*о графика маршрутов?

Вопрос 3. Какой аналог дорожных «пробок» в перевозке пассажиров на городском транспорте?

Упражнение 1. На примере транспортных сайтов проверьте выполнение принципа «динамического программирования» Беллмана. Он гласит, что оптимальный подпуть оптимального пути является оптимальным.

Упражнение 2. Найдите на сайте *www.ptv-vision.com* примеры моделирования дорожно-транспортных ситуаций. Обратите внимание на то, что движение каждой транспортной единицы осуществляется самостоятельно. Сравните с постановкой самостоятельного задания № 3.

3.16. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

3.16.1. Типы данных и ассоциативные массивы

При моделировании дорожно-транспортных задач необходимо создавать типы данных для хранения информации, которая, как правило, поступает из баз данных. Интерфейс приложения может быть реализован на различных языках высокого уровня.

Рассмотрим примеры для языка *PHP*. В языке *PHP* отсутствуют пользовательские типы данных в виде структур или записей (мы здесь не будем говорить о классах на языке *PHP*). Некоторым аналогом структуры является ассоциированный массив. Например, рассмотрим структуру на языке *Си.*

```
struct Person{
        char Fam[20], Im[20];
        struct Date{
                int d, m, y;
        } Birthday;
};
Person P = {“Иванов”, “Иван”, {10, 1, 1990}};
cout << P. Birthday.d; // 10
```

Аналогичная конструкция на языке *PHP* будет иметь вид

```
<?
$P=array (Fam=>"Иванов", Im=>"Иван", Birthday => array(d=>10, m=>1, y => 1990));
```

```
echo $P[Birthday][d]; // 10
?>
```

Массив $*P* является не типом данных, а массивом. Для другого человека придется заново определять ассоциативный массив.

Пример 4. Массив для хранения информации о перекрестках: какие улицы пересекаются, и какие из них являются главными для транспорта.

```
<?
$Perekr[0]= array( array(“улица” => “Ильича”), array(“улица” => “Победы”));
echo $Perecr[0][1][ “улица”]; // Ильича
?>
```

Пример 5. Массив для хранения информации об улице с перечнем перекрестков.

```
<?
$Street[0]= array( $Perecr[0], $Perecr[2]);
?>
```

Пример 6. Заполнить массив $*Street*[“*Ильича*”], перекрестками, содержащими эту улицу.

```
<?
foreach($Perekr as $P)
        foreach($P as $str)
                if($str["улица"]== "Ильича")
                  $Street["Ильича"][] = $P;
echo "Перекрестки улицы Ильича :"."<br>";
for($i=0; $i<count($Street["Ильича"]; $i++)
        for($j=0; $j<count($Street["Ильича"][$i]; $j++)
          echo $Street["Ильича"][$i][$j]["улица"]." - ";
?>
```

Вариант 1. Составьте пользовательский тип данных или массив для хранения следующей информации:

- номер маршрута и последовательность его остановок;
- остановка и перечень проходящих через нее маршрутов с учетом одностороннего или двустороннего направления;
- улица с перечнем остановок на этой улице.

Вариант 2. Составьте пользовательский тип данных или массив для хранения следующей информации:

- геометрия улицы с указанием частей света (юг, север и т. п.) и километража;
- улица и ее достопримечательности с указанием километража;
- площадь и примыкающие к ней улицы.

Вариант 3. Составьте пользовательский тип данных или массив для хранения следующей информации:

- траектория перемещения пешехода по улицам;
- статистика перемещения конкретного пешехода по улицам в течение месяца;
- перечень мест с указанием улицы, которые посетил пешеход в течение дня с указанием времени пребывания.

Вариант 4. Составьте пользовательский тип данных или массив для хранения следующей информации:

- траектория перемещения автомобиля по улицам;
- статистика перемещения конкретного автомобиля в течение месяца;
- перечень мест с указанием улицы, которые посетил автомобилист в течение дня с указанием времени пребывания.

Вариант 5. Составьте пользовательский тип данных или массив для хранения следующей информации:

- светофорный перекресток с указанием всех светофоров и порядка движения по перекрестку при разрешающем сигнале светофора;
- перечень светофорных перекрестков улицы;
- временные интервалы переключения сигналов светофоров на перекрестке;

3.16.2. Поиск информации в файлах

При моделировании дорожно-транспортных задач значительная часть информации содержится не в базах данных, а текстовых файлах. Как правило – это *html*-файлы загружаемых страниц.

Например, поисковые *Internet*-машины анализируют структуру *html*-файлов для определения релевантности найденных сайтов запросам пользователя. В частности, анализируется состав заголовка *<head> </head>*, ключевые слова и т. д.

Пример 7. Откройте сайт *www.ettu.ru*. Выберите страницу с остановками маршрута № 8. Сохраните соответствующий *html*-файл под именем *t*8.*txt*. Этот файл содержит строку «Остановки», после которой идут названия остановок

8-го маршрута. Названия остановок разделены двумя символами: пробел и дефис «– ». Откройте файл *t*8.*txt* и выведите названия остановок на экран.

```
<?
$file = "t8.txt"; // файл с остановками
$fh = fopen($file, "r") or die("File ($file) does not exist!");
$file = fread($fh, 10000);
$regs = split ("Остановки", $file);
$regs = split (" – ", $regs[1]);
foreach ($regs as $q){
        echo $q;
        echo "<br>";
        echo "<br>";
}
fclose($fh);
?>
```

Вариант 1. Откройте сайт *www.ettu.ru.* Выберите страницу с остановками маршрута № 22. Сохраните соответствующий *html*-файл под именем *t*22.*html*. Этот файл содержит строку «Пути следования» с перечнем улиц, по которым едут трамваи маршрута № 22. Откройте файл *t*22.*html* и программным способом прочитайте информацию в массив, который был создан в задании 1.

Вариант 2. Откройте сайт *www.ettu.ru.* Выберите страницу с остановками маршрута № 8. Сохраните соответствующий *html*-файл под именем *t*8.*html*. Этот файл содержит строку «Остановки» с названиями остановок 8-го маршрута. Откройте файл *t*22.*html* и прочитайте информацию об остановках в массив, который был создан в задании 1.

Вариант 3. Откройте сайт *www.transport.ekburg.ru.* Выберите страницу с остановками маршрута № 8. Сохраните соответствующий *html*-файл под именем *t*8.*html*. Этот файл содержит строку «Остановки» с названиями остановок 8-го маршрута. Откройте файл *t*22.*html* и прочитайте информацию об остановках в массив, который был создан в задании 1.

Вариант 4. Откройте сайт *www.transport.ekburg.ru.* Выберите страницу с остановками маршрута № 8. Сохраните соответствующий *html*-файл под именем *t*8.*html*. Этот файл содержит строку «Остановки в обратном направлении» с названиями остановок 8-го маршрута. Откройте файл *t*22.*html* и прочитайте информацию об остановках в массив, который был создан в задании 1.

Вариант 5. Откройте сайт *www.transport.ekburg.ru.* Выберите страницу с остановками маршрута № 8, 17, 22. Прочитайте информацию о конечных

остановках маршрута, длине маршрута, времени в пути. Запишите данные в массив, который был создан в задании 1.

3.16.3. Моделирование дорожно-транспортных ситуаций

Варианты задания п. 3.16.2 позволяют читать/записывать в файл информацию о дорожно-транспортных ситуациях, а в задании п. 3.16.1 определяются пользовательские типы данных для хранения этой информации в памяти компьютера. Теперь, после выполнения этих заданий, можно реализовать моделирование алгоритмов обработки этой информации.

Вариант 1. Допустим, имеется массив маршрутов. Каждый элемент массива содержит номер маршрута и перечень последовательных остановок. Предложите и реализуйте алгоритм поиска маршрута от остановки *A* до остановки *B* с возможными пересадками.

Вариант 2. Имеется однополосная дорога с перекрестками. Светофоры имеют режимы переключения сигналов. В файле записаны координаты и скорости автомобилей, снятые в некоторый момент модельного времени. Постройте модель перемещения автомобилей по дороге. Предусмотрите изменение режима светофоров в соответствии с алгоритмом п. 3.14.

Вариант 3. Имеется абстрактный город, состоящий из прямоугольной сети дорог. Размеры города задаются. Маршруты городского пассажирского транспорта могут проходить по любым улицам. Смоделируйте построение траекторий транспорта, используя алгоритм п. 3.10.1. Для объективной оценки данной транспортной сети вычислите критерии «Покрытие города», изложенный в п. 3.13.2.

Вариант 4. Имеется абстрактный город, состоящий из прямоугольной сети дорог. Размеры города задаются. Маршруты городского пассажирского транспорта могут проходить по любым улицам. Смоделируйте построение транспортных маршрутов, используя алгоритм п. 3.10.1. Для объективной оценки данной транспортной сети вычислите критерии «Количество пересадок», изложенный в п. 3.13.2.

Вариант 5. Имеется абстрактный город, состоящий из прямоугольной сети дорог. Заданы маршруты городского пассажирского транспорта Смоделируйте временные графики маршрутов. Найдите маршрут с наибольшей длительностью.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК К ГЛАВЕ 3

3.1. Насакин Р. Растворители пробок / Р. Насакин // Компьютера. 2007. №29(697). С. 39 – 43.

3.2. Адасинский С.А. Городской транспорт будущего / С.А. Адасинский. М.: Наука, 1979. 235 с.

3.3. Мескон М.Х. Основы менеджмента / М.Х. Мескон, М. Альберт, Ф. Хедоури. М.: Дело, 1992. 346 с.

3.4. Швецов В.И. Математическое моделирование транспортных потоков / В.И. Швецов // Автоматика и телемеханика. 2003. № 11. С. 15 – 18.

3.5. Брайловский Н.О. Моделирование транспортных систем / Н.О. Брайловский, Б.И. Грановский. М.: Транспорт, 1978. 125 с.

3.6. Уизем Дж. Линейные и нелинейные волны / Дж. Уизем. М.: Мир, 1977. 246 с.

3.7. Хейт Ф. Математическая теория транспортных потоков / Ф. Хейт. М.: Мир, 1966. 286 с.

3.8. Смирнов Н.Н. Математическое моделирование автотранспортных потоков / Н.Н. Смирнов, А.Б. Киселев, В.Ф. Никитин. М.: МГУ, 1999. 210 с.

3.9. Алиев А.К. Моделирование транспорта в ИСА РАН / А.К. Алиев, Ю.С. Попков, В.И. Швецов // Компьютерные модели развития города: сб. научных трудов. СПб.: Наука, 2003. С. 78 – 89.

3.10. Усиченко Н.Г. Экономика и организация городского пассажирского транспорта / Н.Г. Усиченко. М.: Финансы и статистика, 2002. 112 с.

3.11. *www.ettu.ru* – сайт ЕМУП «Трамвайно-троллейбусное управление».

3.12. *www.transport.ekburg.ru* – сайт АИС «Транспорт города Екатеринбурга».

3.13. *www.eka.rusavtobus.ru* – сайт «Маршруты Екатеринбурга».

3.14. *www.maps.google.ru* – карты Google.

3.15. *www.tfl.gov.uk* – сайт «Transport of London».

3.16. Hardin G. Tragedy of the commons / G. Hardin // Journal of Heredity. 1959. P. 18.

3.17. Downs Anthony. Still Stuck in Traffic Coping with Peak-Hour Traffic Congestion / Anthony Downs. Brookings Institution Press, 2004. 455 p.

3.18. *www.tram.ruz.net* – сайт «Московский трамвай».

3.19. Трофимов С.П. Моделирование и оптимизация маршрута городского электротранспорта для пассажира / С.П. Трофимов, М.Ю. Низова, Н.Г. Дружинина, О.Г. Трофимова // Научные труды V Международной научно-практической конференции «СВЯЗЬ-ПРОМ 2008». Екатеринбург: ЗАО «Компания Реал-Медиа», 2008. С. 302 – 304.

3.20. Печерский М.П. Автоматизированные системы управления дорожным движением в городах / М.П. Печерский, Б.Г. Хорович. М.: Транспорт, 1979. 176 с.

3.21. Артыков А.П. Автоматизация управления уличным пассажирским транспортом в больших городах за рубежом / А.П. Артыков. М.: ГОСИНГИ, 1975. 39 с.

3.22. Рушевский П.В. Организация и регулирование уличного движения с применением автоматических средств управления / П.В. Рушевский. М.: Высшая школа, 1974. 238 с.

3.23. Григорьев Д.А. Свидетельство об официальной регистрации программы для ЭВМ «Нейросетевая система управления потоками транспорта на перекрестках произвольной конфигурации» / Д.А. Григорьев, С.И. Шаймарданова, Т.С. Балыклов. № 2002611009 от 20.06.2002 г. М.: Роспатент.

3.24. Юсупова Н.И. Модели и алгоритмы управления потоком транспорта на основе нечеткой логики / Н.И. Юсупова, Д.Н. Бажин // Российская научно-методическая конференция с международным участием «Управление экономикой: методы, модели, технологии». Сб. научных трудов в 3-х частях. Ч. 2. Уфа: УГАТУ, 2001. С. 19 – 24.

4. ПРИМЕР ПРОГРАММНОЙ РЕАЛИЗАЦИИ ИНФОРМАЦИОННО-КОММУНИКАЦИОННОЙ ТРАНСПОРТНОЙ СИСТЕМЫ

4.1. ИНФОРМАЦИОННО-КОММУНИКАЦИОННАЯ СИСТЕМА МУП ТТУ Г. ЕКАТЕРИНБУРГА

Информационно-коммуникационной системы МУП ТТУ г. Екатеринбурга (рис. 4.1) включает в себя следующие взаимосвязанные программные комплексы: «Составление расписания маршрутизированного транспорта», «Подготовка нарядов водителей и кондукторов», «Диспетчер выпуска и движение подвижной единицы (ПЕ)», «Табель учета рабочего времени водителей и кондукторов», «Путевой лист автотранспортной службы МУП ЕТТУ», «Транспорт города Екатеринбурга» [4.1].

Рис. 4.1. Модель информационно-коммуникационной системы ЕМУП ТТУ

Данная система позволяет получить отчетные технико-экономические показатели о работе ПЕ, водителя и кондуктора, структурных подразделений и всего предприятия в целом. Для реализации информационно-коммуникационной системы на *web*-сервере разработана единая база данных, позволяющая оперативно и полно интегрировать данные из одного модуля в другой, не дублируя информацию.

Программный комплекс «Составление расписания движения маршрутизированного транспорта» позволяет решить многокритериальную задачу с учетом следующих параметров: день недели, конечные станции, промежуточные контрольные пункты, остановочные пункты, участки маршрутной сети, наряд выпуска подвижного состава. В результате создается электронная визуализированная маршрутная сеть с автоматизированной корректировкой расписания, а также производится расчет технико-экономических показателей планируемой работы подвижного состава, маршрутов, водителей и предприятия в целом.

Программный комплекс «Подготовка нарядов водителей и кондукторов» предназначен для составления и ведения нарядов на работу водителей и кондукторов, подвижной единицы (ПЕ) маршрутизированного транспорта.

Программный комплекс «Диспетчер выпуска и движение подвижной единицы» осуществляет контроль над выпуском на линию запланированного числа технически исправных подвижных единиц (ПЕ), согласно расписанию выхода ПЕ и наряду работы водителей и кондукторов, а также производит своевременную оперативную замену ПЕ как на линии, так и в депо из ПЕ, находящихся в резерве. Диспетчер выпуска следит за временем начала работы водителей в депо, оформляет путевой лист на ПЕ. Все изменения в путевом листе водителя и кондуктора или в базовом наряде ПЕ оперативно вносятся в данный программный комплекс.

Программный комплекс «Путевой лист автотранспортной службы ЕМУП ТТУ» позволяет сформировать показатели работы автотранспортной службы в целом, основываясь на информации для каждой автомашины: общий и дополнительной пробег, фактический расход топлива, экономия или перерасход топлива.

Программный комплекс «Транспорт города Екатеринбурга» реализован в виде веб-сайта и предоставляет населению информацию о маршрутах и расписании движения трамваев и троллейбусов в г. Екатеринбурге, выполнении плана, справку о сходе трамваев и троллейбусов, сведения о движении поездов на линии по маршрутам, задержках движения на линии и в депо по техническим неисправностям, показателях работы ЕМУП ТТУ. Данный сайт позволяет Комитету по транспорту и организации дорожного движения осуществлять ежедневный контроль выполнения основных показателей ЕМУП ТТУ, получать итоговые данные выполнения муниципального заказа по показателям работы ЕМУП ТТУ за период (за любой день, за любой месяц, за любой год), проводить ежедневный и ежемесячный контроль за регулярностью

движения по маршрутам трамваев и троллейбусов, знать о выпуске подвижного состава на улицы города в любой заданный момент. Оперативность данных позволяет наиболее полно информировать население об изменениях в движении трамваев и троллейбусов, как постоянных, так и временных.

Специальные условия применения и требования организационного, технического и технологического характера следующие. Информационно-коммуникационная система реализована в современной архитектуре клиент-сервер, причем серверная часть функционирует в операционной среде *Windows NT* или *LINUX* с использованием СУБД *MySQL*, а клиентские места работают под управлением *Windows* 9*x, Windows NT* и выше. Для работы с базой данных программного комплекса на сервере должна быть установлена СУБД *MySQL* (версия не ниже 4.0.17), а на клиентской машине – *BDE Administrator Delphi* 3.0. Для написания отчетов на сервере устанавливаются: файл-сервер – *SAMBA* и инструмент для написания отчетов (*PHP* версия не ниже 4.3.3). Для просмотра отчетов на сервере устанавливается *Web: Apache Version Apache*/2.0.47 (*Unix*). Пользовательский интерфейс на клиентской машине – любой обозреватель: *Microsoft Internet Explorer* (версия не ниже 5.0), *Netscape Communicator* (версия не ниже 4.51) или др. Для устойчивой работы системы достаточно персонального компьютера с процессором *Intel Pentium* I и выше. Оперативной памяти на персональном компьютере для работы в системе должно быть не менее 256 Мб.

Программные комплексы «Составление расписания движения маршрутизированного транспорта», «Диспетчер выпуска и движение подвижной единицы», «Путевой лист автотранспортной службы ЕМУП ТТУ», «Транспорт города Екатеринбурга» включены в государственный банк данных и зарегистрированы в Отраслевом фонде алгоритмов и программ, а также получены государственные регистрации разработок в «Национальном информационном фонде неопубликованных документов» и свидетельства на разработку [4.2] – [4.9].

4.2. ПРОГРАММНЫЙ КОМПЛЕКС «СОСТАВЛЕНИЕ РАСПИСАНИЯ МАРШРУТИЗИРОВАННОГО ТРАНСПОРТА» В СРЕДЕ DELPHI

Программный комплекс «Составление расписания маршрутизированного транспорта» включает в себя составление расписания движения трамваев и троллейбусов, с возможностью формирования на любые дни недели, на конечные станции, промежуточные контрольные пункты, остановочные пункты и для каждого наряда (поездные), предусматривающее:

- автоматический ввод данных для расчета;
- работу с электронной маршрутной схемой;
- составление и редактирование расписания;
- автоматизированную корректировку поездных расписаний одновременно с маршрутным расписанием;

- увязку расписаний по участкам маршрутной сети;
- формирование наряда выпуска подвижного состава в целом по сети, в т.ч. и развернутого наряда по маршрутам, графикам и сменам.

Информация предоставляется в табличном и графическом виде.

После составления расписания маршрутизированного транспорта производится расчет технико-экономических показателей планируемой работы подвижного состава, маршрутов, водителей и предприятия в целом, в том числе:

- количество рейсов по каждому наряду, парку, маршруту и в целом по сети;
- наличие подвижного состава по часам суток по каждому маршруту и в целом по сети;
- интервалы движения, вагоно-часы, вагоно-километры.

Правильно составленные с помощью данной программы расписания обеспечивают выполнение основных плановых показателей по регулярности движения пассажирского транспорта, перевозке пассажиров и выручке.

Программный комплекс «Составление расписания маршрутизированного транспорта» выполняет следующие функций:

- хранение набора расписаний;
- создание нового расписания (в том числе на основе одного из предыдущих);
- просмотр и модификация исходных данных для составления расписания;
- автоматическое формирование поездного расписания;
- автоматический расчет плановых показателей;
- составление групп для наряда;
- формирование справок и дополнительных отчетов;
- подготовка итоговых выходных данных для использования другими программами.

Расчет технико-экономических показателей обеспечивает возможность формирования действующих и создания новых выходных форм в соответствии с требованиями предприятия, корректировки расписаний движения маршрутов по конечным станциям, контрольным пунктам, остановкам и поездных расписаний. Реализована возможность просмотра и печати всех формируемых документов.

На рис. 4.2. представлена концептуальная модель «Составление расписания маршрутизированного транспорта».

Для определения пробегов между остановками рассчитываются нормы пробега N (4.1) с учетом следующих параметров:

- времени посадки-высадки пассажиров на остановочных пунктах, T_p;
- межперегонного пробега с учетом правил дорожного движения подвижных единиц, T_{mp};
- сезонных погодных явлений (дождь, юз, снегопад, листопад и т.п.), K_c;
- дорожной обстановки (закрытия на ремонт, "пробки"), K_o;
- состояния покрытия дорожного участка, K_d;
- времени суток, K_v:

$$N = T_p + T_{mp}\, K_c\, K_o\, K_d\, K_v. \qquad (4.1)$$

Рис. 4.2. Концептуальная модель «Составление расписания маршрутизированного транспорта»

Для каждого графика маршрута по (4.2) рассчитываются поездо-километры *LPKM*, с учетом следующих параметров:

- количество рейсов в прямом (нулевом) направлении движения маршрута, *KolR*0;

- количество рейсов в обратном направлении движения маршрута, *KolR*1;
- длина рейса в прямом направлении, *dlR*0;
- длина рейса в обратном направлении, *dlR*1;
- длина рейса при выходе из депо *LNulV*;
- длина рейса при заходе в депо, *LNulZ*;
- длина спецрейсов, *LSpecReys*:

$$LPKM = KolR0\ dlR0 + KolR1\ dlR1 + LnulV + LnulZ + LspecReys. \quad (4.2)$$

Упрощенная схема базы данных «Составление расписания маршрутизированного транспорта» представлена на рис. 4.3.

Функции программного комплекса по составлению расписания маршрутизированного транспорта:

- открытие расписания;
- закрытие расписания;
- создание расписания;
- переименование расписания;
- удаление расписания;
- копирование маршрутов и графиков между расписаниями:
  - выбор расписаний;
  - копирование графика;
  - удаление графика;
- описание маршрутной сети:
  - ввод, удаление и редактирование остановок;
  - ввод, удаление и редактирование связок по длине между остановками;
  - ввод, удаление и редактирование связок по времени между остановками;
  - описание, удаление и редактирование конфигурации маршрутов;
  - описание, удаление и редактирование конфигурации специальных маршрутов;
- составление и редактирование расписания:
  - редактирование графиков выхода;
  - редактирование режимов работы графиков;
  - загрузка составленного расписания;
  - составление расписания;
  - редактирование:
    - тур влево;
    - тур вправо;

- изменение маршрута влево;
- изменение маршрута вправо;
- редактирование записей расписания;
- просмотр таблицы поездного расписания;
- просмотр таблицы маршрутного расписания;
- просмотр плановых показателей по смене, графику, маршруту, депо;
- выбор способов и масштаба отображения расписания;
- выбор отображаемых остановок;
- «стуки», т.е. одновременный проезд двух поездов через один контрольный пункт;

- составление и редактирование групп для наряда;
- просмотр плановых показателей;
- редактирование справочника режимов работы графиков;
- редактирование справочника подготовительно-заключительного времени;
- тестирование маршрутной сети на наличие ошибок.

Для каждого вида транспорта формируется своя схема движения. Выбрав маршрут, можно увидеть его на схеме (рис. 4.4).

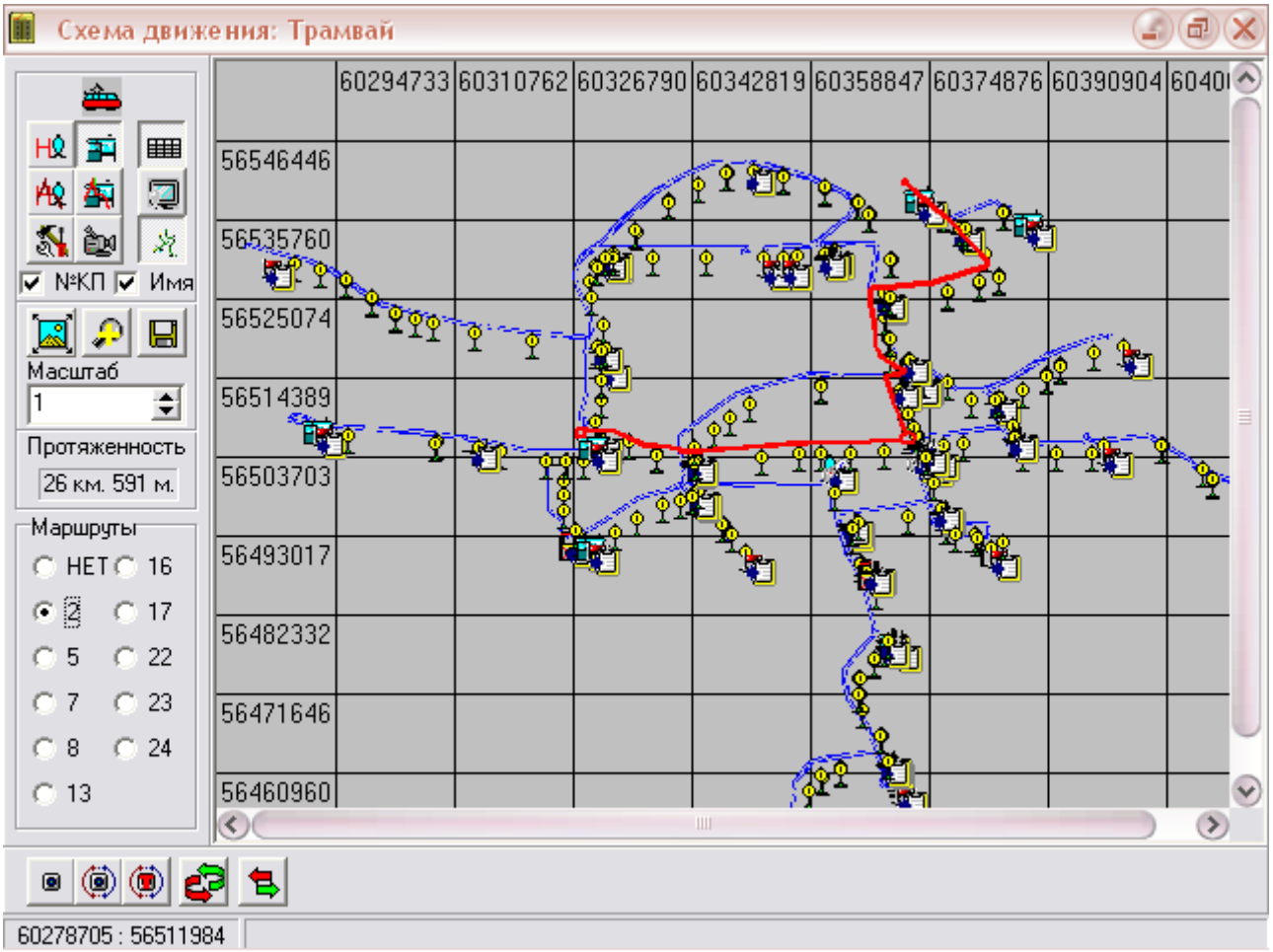


Рис. 4.4. Маршрутная схема движения трамвая

Панель управления электронной схемы находится в левой части формы и предназначена для изменения режимов просмотра схемы.

На рис. 4.5 представлен список маршрутов и графиков (левый нижний угол окна). Галочкой помечены те графики движения, расписания которых отображены на рис. 4.5. В нижней части окна расположена шкала, где выбирается временной интервал между контрольными точками маршрута. При передвижении курсора по рабочей зоне окна программы, в верхней информационной строке можно наблюдать время, которое соответствует местонахождению курсора. Таким образом, чтобы узнать время прибытия графика на конечную станцию, необязательно включать режим отображения времени, а достаточно подвести стрелку курсора к заданной точке. После выбора маршрутов окно редактирования будет иметь вид, представленный на рис. 4.5.

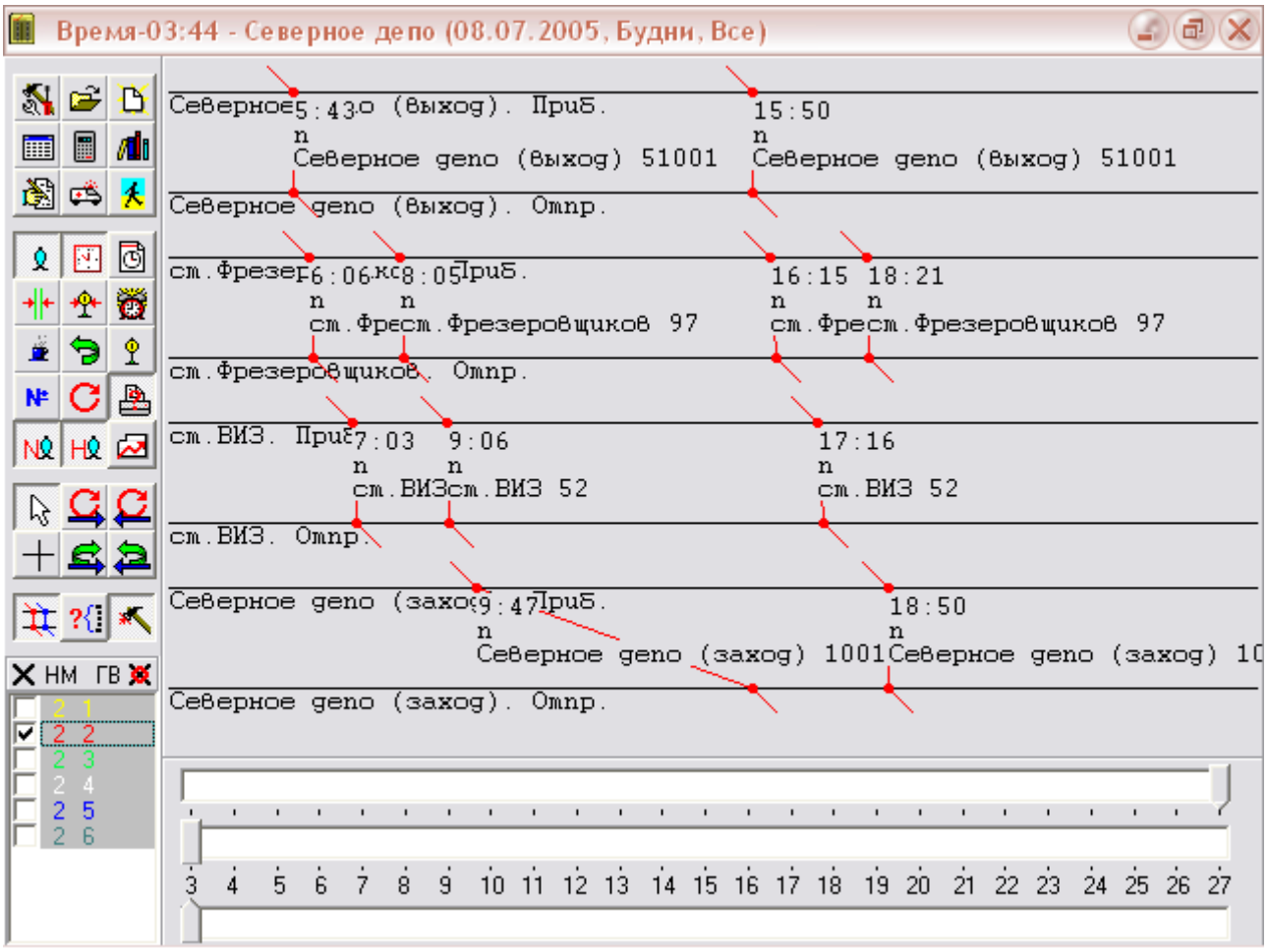


Рис. 4.5. Редактирование расписания

В режиме редактирования можно осуществлять следующие операции.

- *Изменение туров*. Для этого служат кнопки «Тур влево» и «Тур вправо» на панели редактирования расписания. После того как активизировалась одна из этих кнопок, нужно щелкнуть по точке, соответствующей прибытию на конечную, и на графике либо добавится, либо удалится рейс в зависимости от того, какая кнопка использовалась – «Тур влево» или «Тур вправо».

- *Изменение маршрутов*. Эта операция используется при изменении конфигурации отдельного графика на маршруте. Для этого выбирается одна из конечных на маршруте и предполагается, что с этой конечной поезд идет в депо. Если выбрать первую конечную по прибытии на маршрут, то меняется нулёвка (выход из депо) прибытия, а если любая другая, то меняется нулёвка отправления (заезд в депо).

- *Перемещение графиков*. Использование этой операции дает возможность изменить время пробега от одной конечной до другой в зависимости от времени суток, дней недели и т.д. Для этого надо выбрать кнопку «Перемещение графиков» на панели инструментов окна редактирования расписания и на графике выбрать искомую конечную остановку и переместить её по временной шкале в другое место. В результате изменится весь график движения подвижной единицы.

Кроме редактирования путем перемещения точек на графике, соответствующих основным остановкам, можно осуществлять корректировку путем ввода данных в поля ввода информации (рис. 4.6).

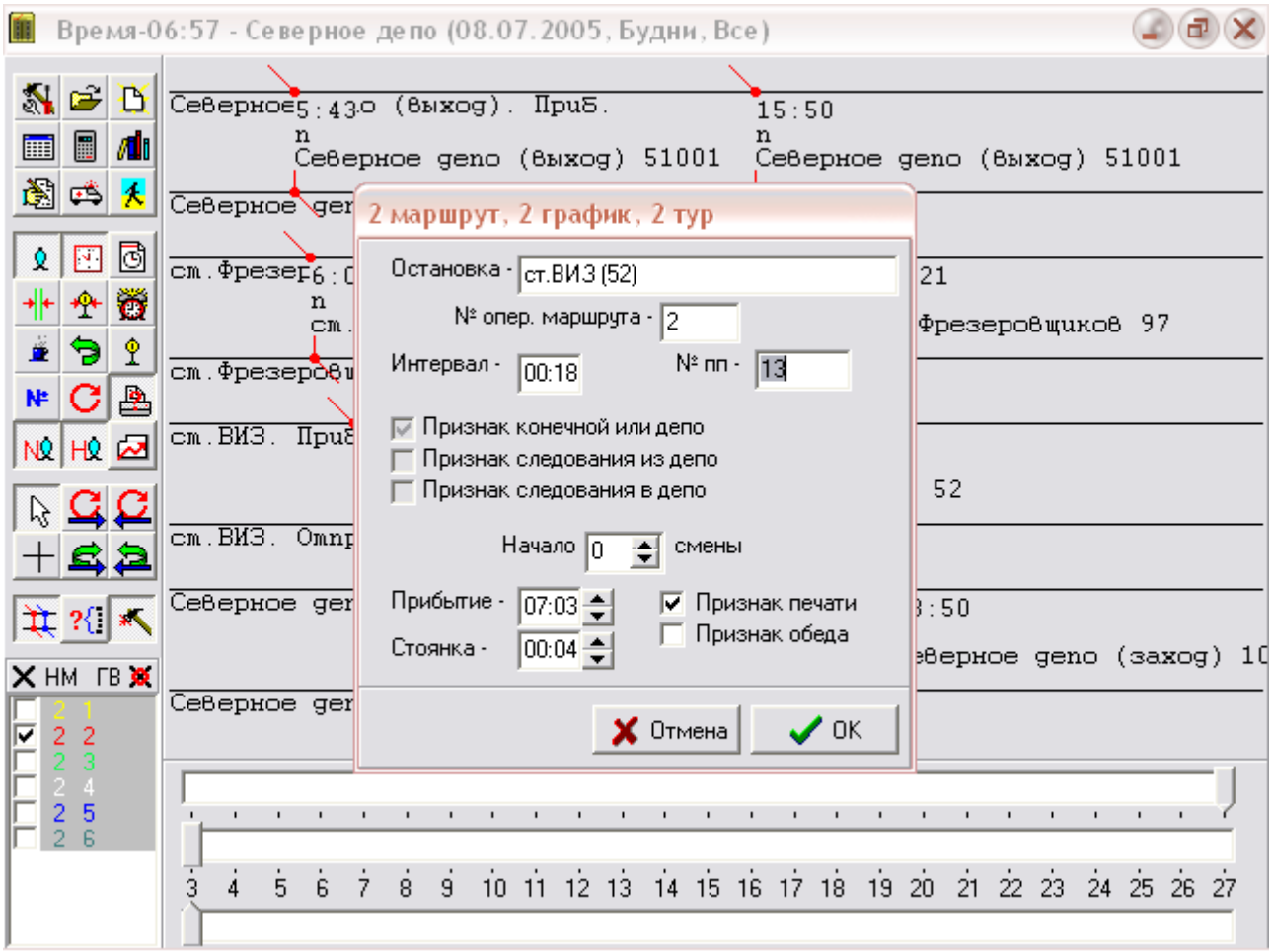


Рис. 4.6. Оперативное окно для ввода информации

Для этого нужно щелкнуть мышью по точке, соответствующей остановке. После этого выйдет оперативное окно для ввода информации.

На основе составленного расписания автоматически рассчитываются плановые показатели (рис. 4.7) по каждой смене, по графикам, по маршрутам и в целом по депо.

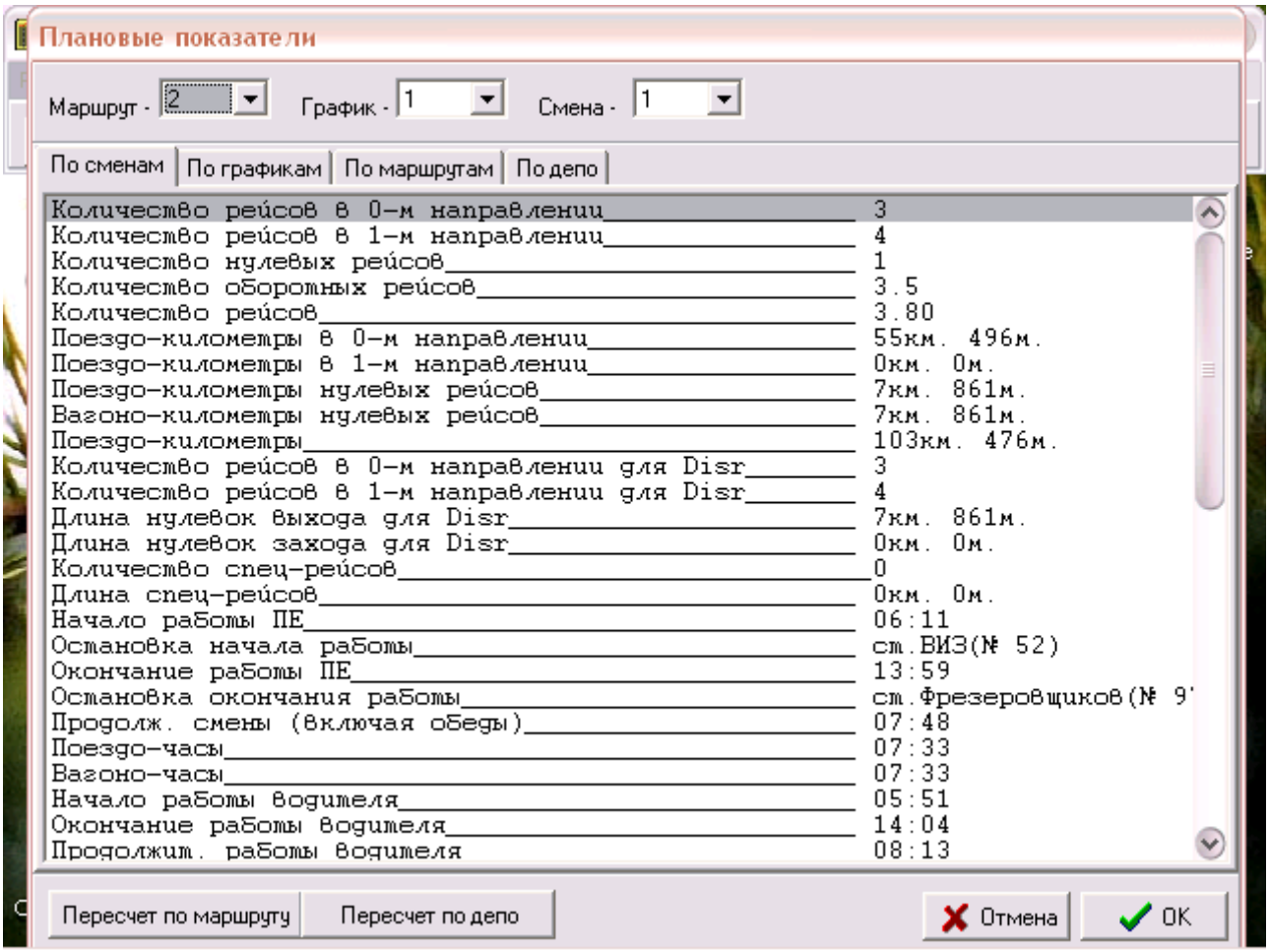


Рис. 4.7. Расчет плановых показателей

Исходные данные программного комплекса «Составление расписания маршрутизированного транспорта»:

- список видов транспорта;
- список транспортных предприятий;
- список остановок различного типа (в том числе с признаками «Составление расписания», «Конечная», «Депо») по виду транспорта;
- граф пробегов между остановками (длины в метрах) по виду транспорта;
- список времени пробега между остановками с признаком «Составление расписания» в соответствии с:

  - видом транспорта;
  - сезоном (зима, лето);
  - временем суток;
  - проходящими маршрутами;

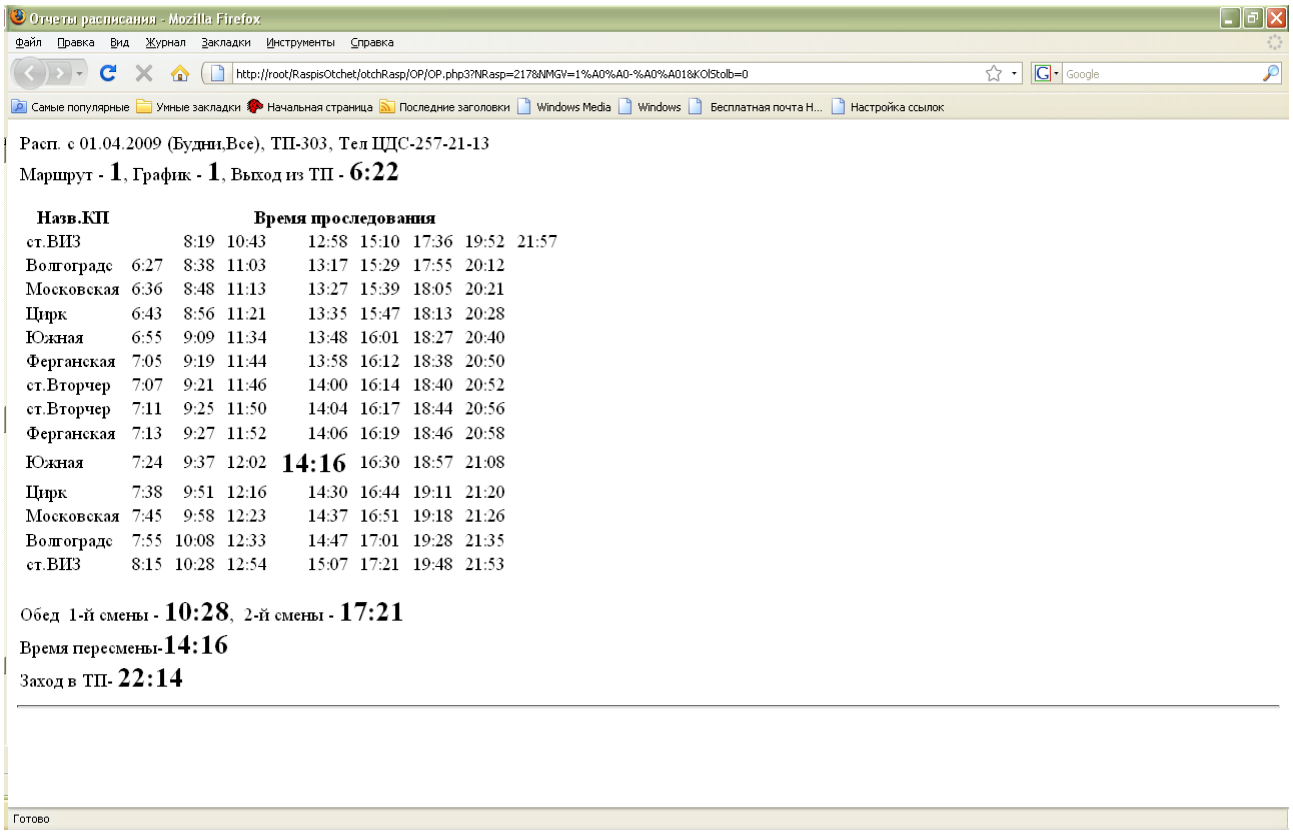
- список, типы и конфигурация (последовательность остановок) маршрутов и специальных маршрутов по депо;
- список и режимы работы графиков маршрута;
- описание графиков выхода маршрута:

  - количество графиков выхода в маршруте;
  - №, КП и время прибытия первого графика выхода;
  - примерное время окончания выходов;
  - время начала, продолжительность и КП обедов по сменам;
  - примерное время начала и окончания заходов;
  - список специальных маршрутов убытия и прибытия на маршрут;

- информация о количестве групп для наряда по маршруту и количеству графиков в них;
- подготовительно-заключительное время по депо и режиму работы графиков.

В результате работы данного программного комплекса можно получить:

- расписание движения маршрутизированного транспорта;
- полный список плановых показателей по смене, графику, маршруту, депо;
- формы установленных отчетов, например рис. 4.8.



Расп. с 01.04.2009 (Будни,Все), ТП-303, Тел ЦДС-257-21-13
Маршрут - **1**, График - **1**, Выход из ТП - **6:22**

Назв.КП	Время проследования							
ст.ВПЗ		8:19	10:43	12:58	15:10	17:36	19:52	21:57
Волгоградс	6:27	8:38	11:03	13:17	15:29	17:55	20:12	
Московская	6:36	8:48	11:13	13:27	15:39	18:05	20:21	
Цирк	6:43	8:56	11:21	13:35	15:47	18:13	20:28	
Южная	6:55	9:09	11:34	13:48	16:01	18:27	20:40	
Ферганская	7:05	9:19	11:44	13:58	16:12	18:38	20:50	
ст.Вторчер	7:07	9:21	11:46	14:00	16:14	18:40	20:52	
ст.Вторчер	7:11	9:25	11:50	14:04	16:17	18:44	20:56	
Ферганская	7:13	9:27	11:52	14:06	16:19	18:46	20:58	
Южная	7:24	9:37	12:02	**14:16**	16:30	18:57	21:08	
Цирк	7:38	9:51	12:16	14:30	16:44	19:11	21:20	
Московская	7:45	9:58	12:23	14:37	16:51	19:18	21:26	
Волгоградс	7:55	10:08	12:33	14:47	17:01	19:28	21:35	
ст.ВПЗ	8:15	10:28	12:54	15:07	17:21	19:48	21:53	

Обед 1-й смены - **10:28**, 2-й смены - **17:21**
Время пересмены-**14:16**
Заход в ТП- **22:14**

Рис. 4.8. Расписание движения ПЕ через контрольные пункты

4.3. ПРОГРАММНЫЙ КОМПЛЕКС «ПОДГОТОВКА НАРЯДОВ ВОДИТЕЛЕЙ И КОНДУКТОРОВ МАРШРУТИЗИРОВАННОГО ТРАНСПОРТА» В СРЕДЕ DELPHI

Программный комплекс «Подготовка нарядов водителей и кондукторов» предназначен для составления и ведения нарядов на работу водителей и кондукторов, подвижной единицы маршрутизированного транспорта (рис. 4.9).

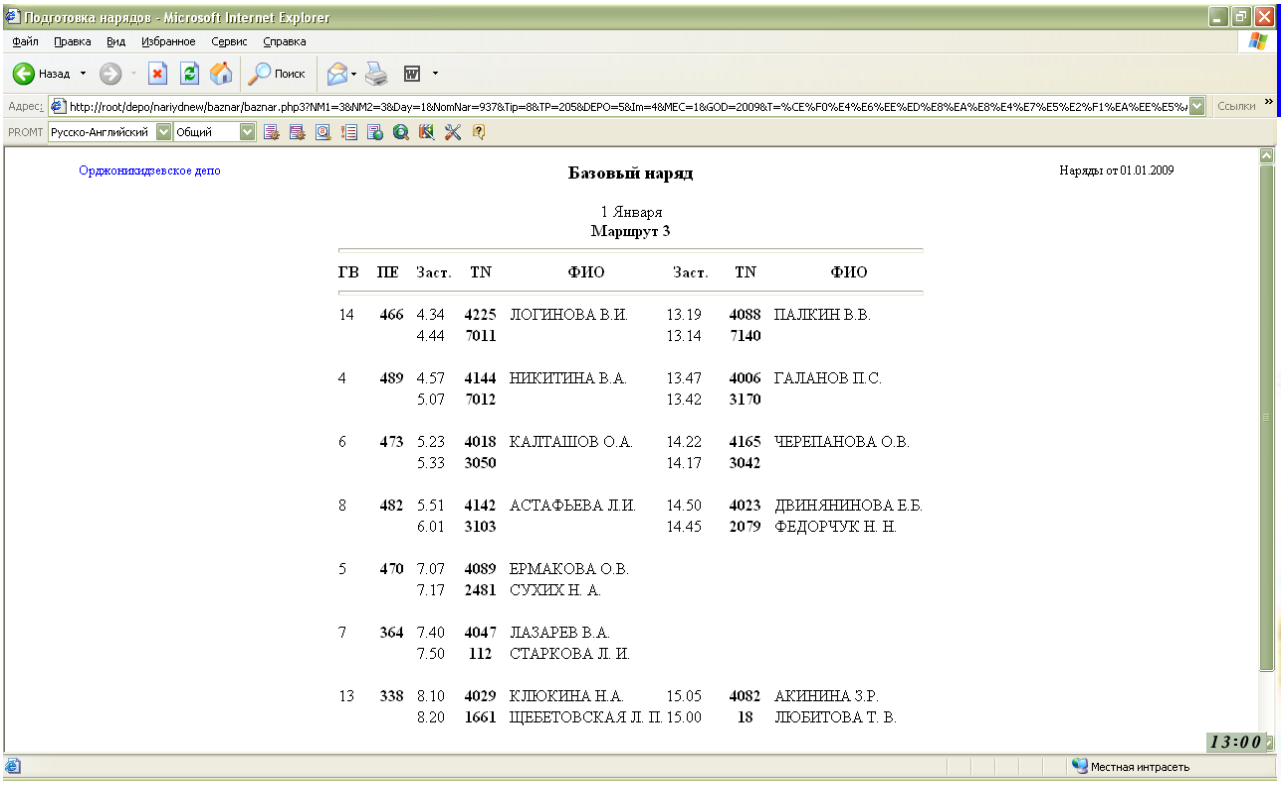


Рис. 4.9. Базовый наряд ПЕ на маршрут

На рис. 4.10 представлена концептуальная модель «Подготовка нарядов водителей и кондукторов».

Упрощенная схема базы данных «Подготовка нарядов водителей и кондукторов» и «Табель учета рабочего времени водителей и кондукторов» представлена на рис. 4.11.

Работа нарядчика транспортного предприятия включает в себя:

1) формирование наряда маршрута (плана работы водителя и кондуктора на месяц) (см. рис. 4.12):

- группировка ПЕ (см. рис. 4.13);
- закрепление ПЕ за маршрутом и группой;
- закрепление водителя за ПЕ;
- закрепление смены за водителем (кондуктором);
- корректировка рабочего времени водителя (кондуктора) в соответствии с нормой;

- подготовка наряда на последующие дни для диспетчера выпуска с учетом внесенных изменений;

2) формирование справок и дополнительных отчетов по нарядам на работу водителей и кондукторов;

3) подготовка итоговых выходных данных для использования другими программами;

Рис. 4.10. Концептуальная модель «Подготовка нарядов водителей и кондукторов»

4) ежедневная корректировка наряда (см. рис. 4.14):

- замена ПЕ;
- замена водителя (кондуктора);
- замена смен;
- замена графиков выхода.

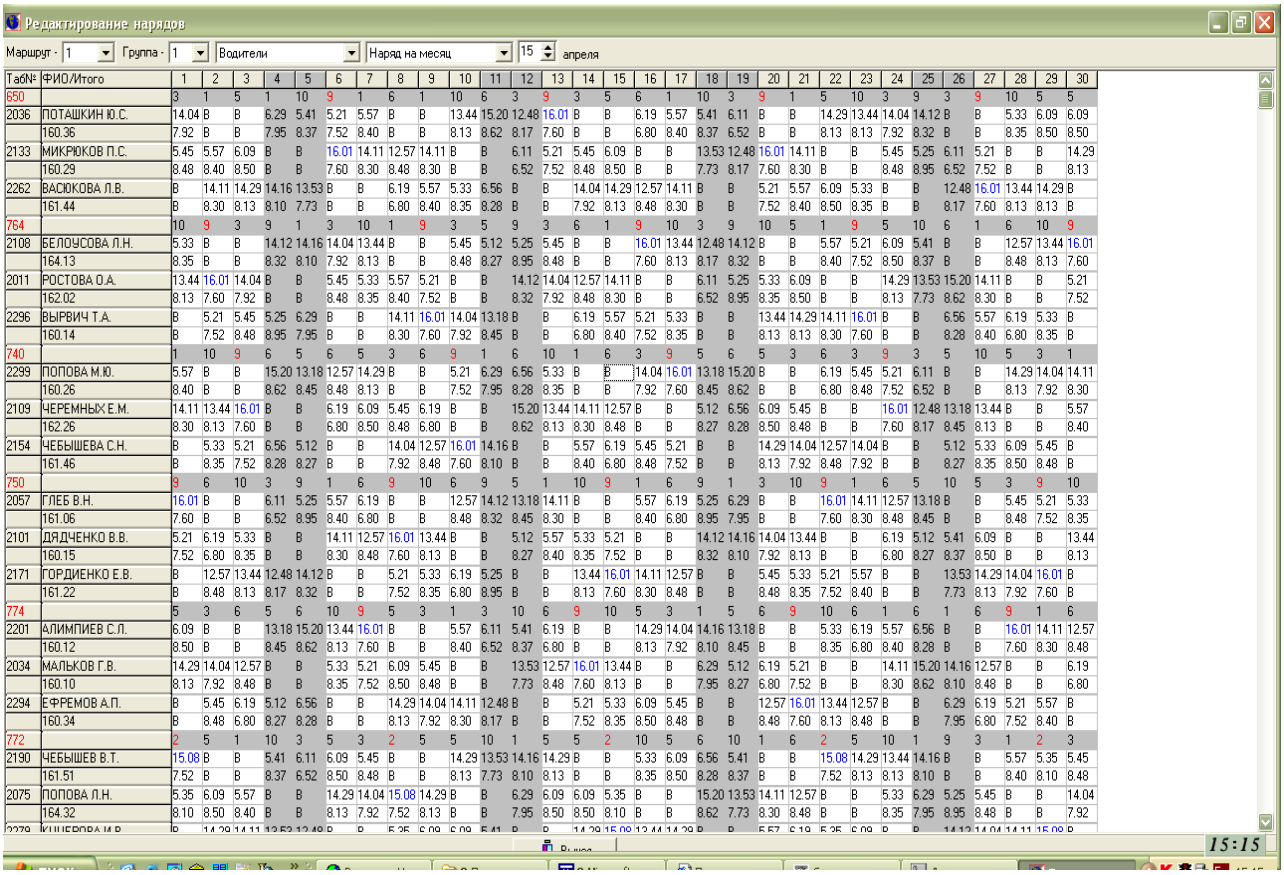
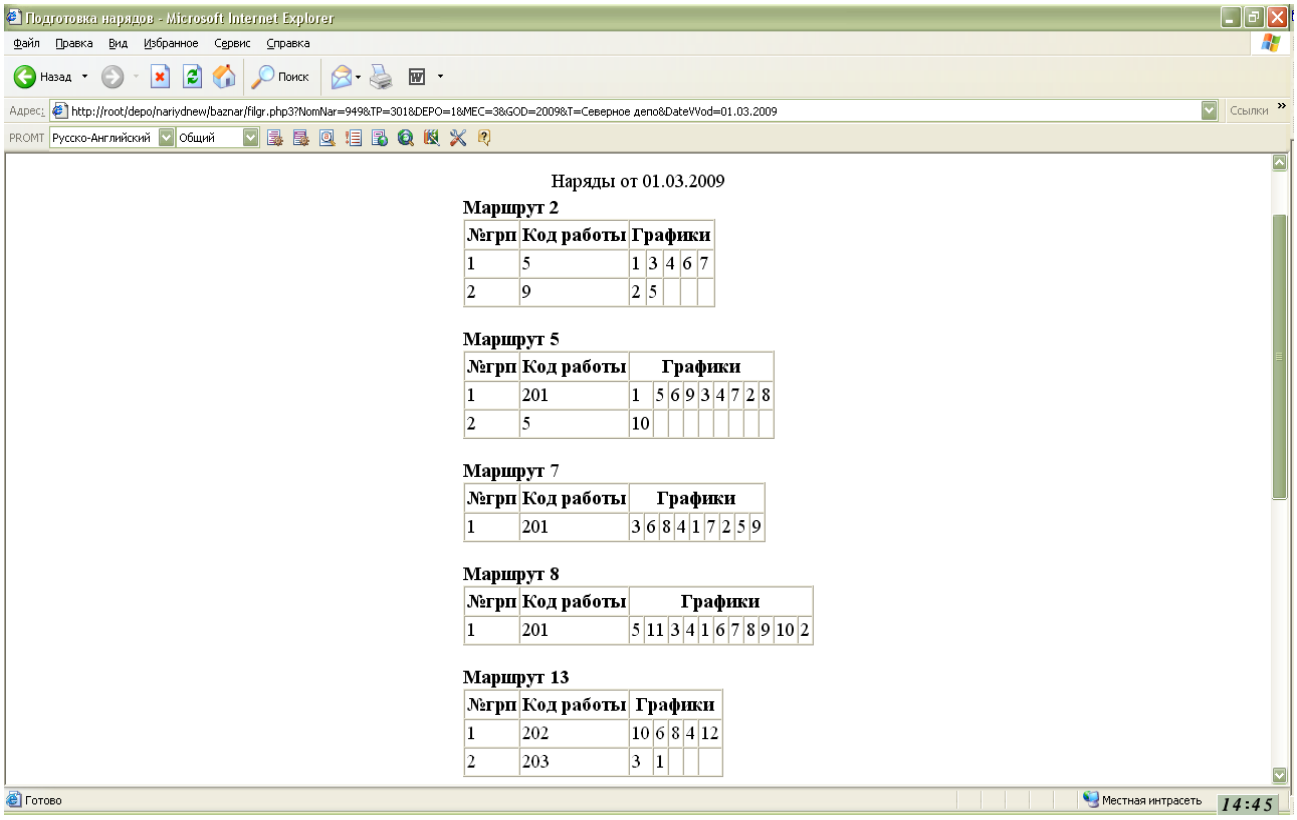


Рис. 4.12. Формирование наряда маршрута



Наряды от 01.03.2009

Маршрут 2

№грп	Код работы	Графики
1	5	1 3 4 6 7
2	9	2 5

Маршрут 5

№грп	Код работы	Графики
1	201	1 5 6 9 3 4 7 2 8
2	5	10

Маршрут 7

№грп	Код работы	Графики
1	201	3 6 8 4 1 7 2 5 9

Маршрут 8

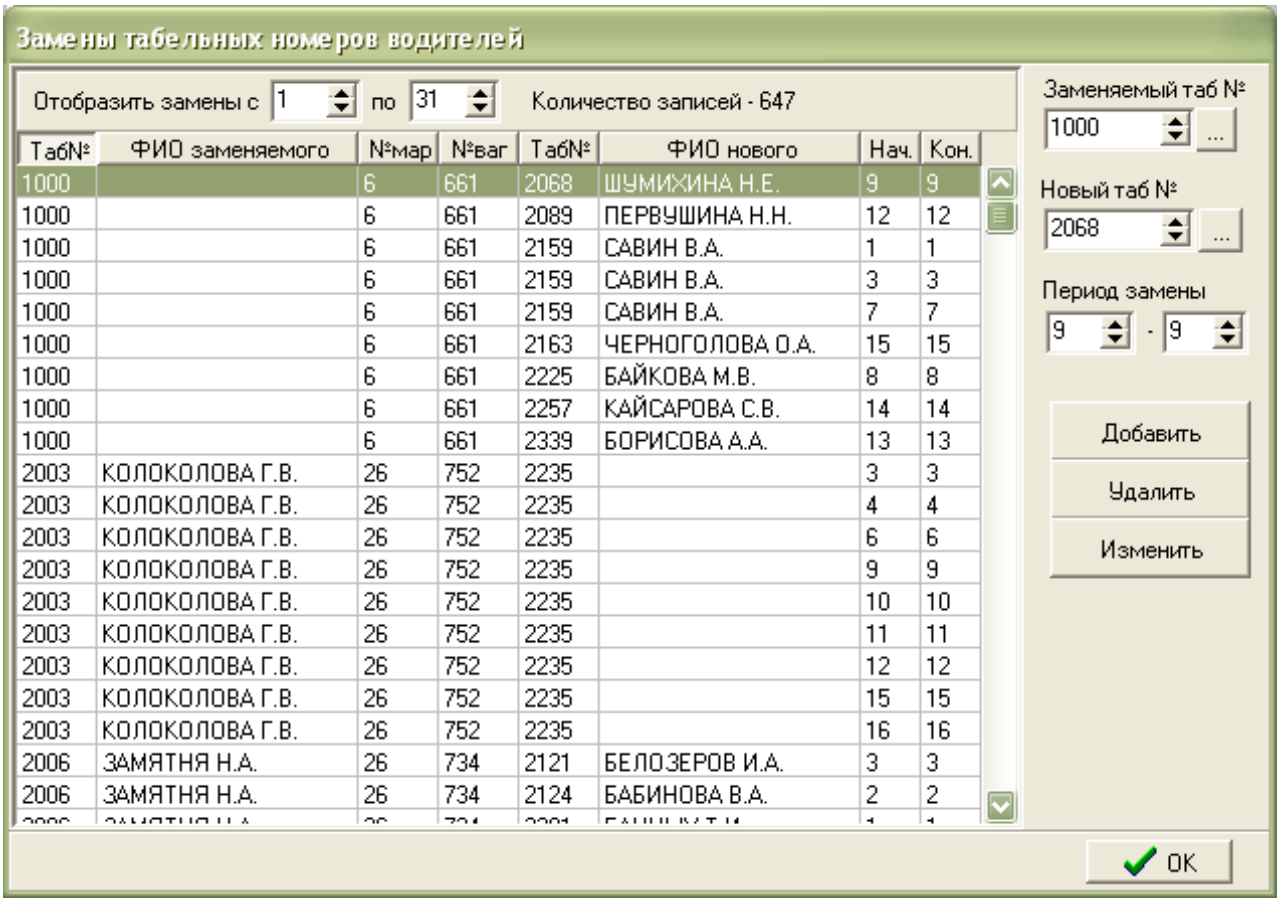
№грп	Код работы	Графики
1	201	5 11 3 4 1 6 7 8 9 10 2

Маршрут 13

№грп	Код работы	Графики
1	202	10 6 8 4 12
2	203	3 1

Рис. 4.13. Группировка ПЕ

Исходными данными для формирования базовых нарядов являются расписание движения ПЕ, закрепление графиков за вагонами, закрепление номеров смен и табельных номеров водителей и кондукторов, ФИО водителей и кондукторов, изменения расстановки графиков, изменения закреплений табельных номеров и смен водителей, справочники дней осмотра ПЕ, режимов работы графиков, кодов отсутствия водителей, отработанное время работы водителей.



Таб№	ФИО заменяемого	№мар	№ваг	Таб№	ФИО нового	Нач.	Кон.
1000		6	661	2068	ШУМИХИНА Н.Е.	9	9
1000		6	661	2089	ПЕРВУШИНА Н.Н.	12	12
1000		6	661	2159	САВИН В.А.	1	1
1000		6	661	2159	САВИН В.А.	3	3
1000		6	661	2159	САВИН В.А.	7	7
1000		6	661	2163	ЧЕРНОГОЛОВА О.А.	15	15
1000		6	661	2225	БАЙКОВА М.В.	8	8
1000		6	661	2257	КАЙСАРОВА С.В.	14	14
1000		6	661	2339	БОРИСОВА А.А.	13	13
2003	КОЛОКОЛОВА Г.В.	26	752	2235		3	3
2003	КОЛОКОЛОВА Г.В.	26	752	2235		4	4
2003	КОЛОКОЛОВА Г.В.	26	752	2235		6	6
2003	КОЛОКОЛОВА Г.В.	26	752	2235		9	9
2003	КОЛОКОЛОВА Г.В.	26	752	2235		10	10
2003	КОЛОКОЛОВА Г.В.	26	752	2235		11	11
2003	КОЛОКОЛОВА Г.В.	26	752	2235		12	12
2003	КОЛОКОЛОВА Г.В.	26	752	2235		15	15
2003	КОЛОКОЛОВА Г.В.	26	752	2235		16	16
2006	ЗАМЯТНЯ Н.А.	26	734	2121	БЕЛОЗЕРОВ И.А.	3	3
2006	ЗАМЯТНЯ Н.А.	26	734	2124	БАБИНОВА В.А.	2	2

Рис. 4.14. Ежедневная корректировка наряда

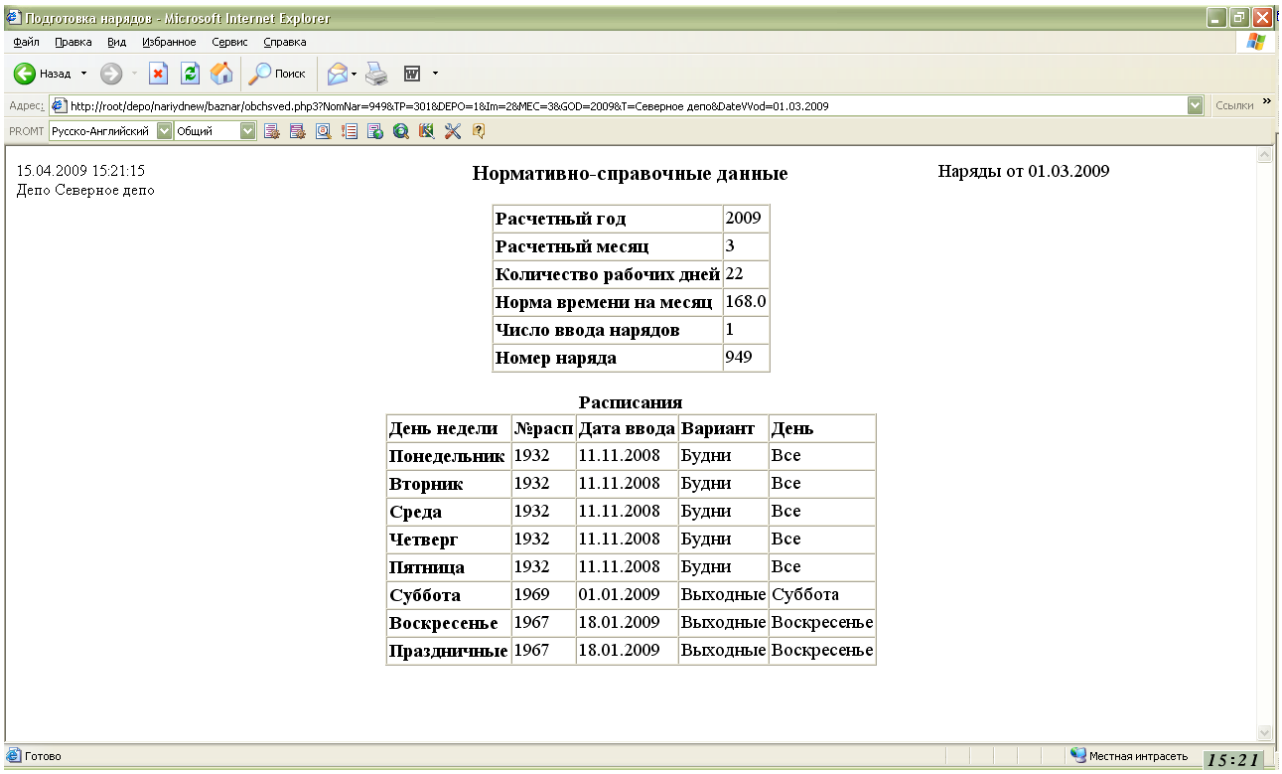
Программный комплекс «Подготовка нарядов водителей и кондукторов» позволяет автоматизировать следующие основные функции нарядчика:

- формирование основных групп ПЕ для проведения еженедельного технического осмотра каждой;
- закрепление смен на каждый день месяца за водителем (кондуктором);
- определение количества рабочих дней за месяц;
- подсчет ежедневного рабочего времени и отработанного за месяц;
- закрепление графика выхода за ПЕ с учетом позднего окончания второй смены;
- равномерное распределение ночных смен водителей (кондукторов);
- чередование графиков заступления водителей (кондукторов);

- чередование утреннего заступления водителей (кондукторов);
- анализ графика работы предыдущего дня;
- выравнивание заложенного по наряду рабочего времени водителей (кондукторов) за месяц и планового;
- закрепления за ПЕ водителей и кондукторов на текущий день;
- оперативный поиск водителя или кондуктора для замены отсутствующих.

Исходные данные программного комплекса «Подготовка нарядов водителей и кондукторов»:

- нормативно-справочные данные (рис. 4.15);



Расчетный год	2009
Расчетный месяц	3
Количество рабочих дней	22
Норма времени на месяц	168.0
Число ввода нарядов	1
Номер наряда	949

Расписания

День недели	№расп	Дата ввода	Вариант	День
Понедельник	1932	11.11.2008	Будни	Все
Вторник	1932	11.11.2008	Будни	Все
Среда	1932	11.11.2008	Будни	Все
Четверг	1932	11.11.2008	Будни	Все
Пятница	1932	11.11.2008	Будни	Все
Суббота	1969	01.01.2009	Выходные	Суббота
Воскресенье	1967	18.01.2009	Выходные	Воскресенье
Праздничные	1967	18.01.2009	Выходные	Воскресенье

Рис. 4.15. Нормативно-справочные данные

- список водителей;
- список подменных водителей;
- список кондукторов;
- список подменных кондукторов;
- закрепление водителей за ПЕ;
- закрепление кондукторов за ПЕ;
- закрепление ПЕ за маршрутом и группой;
- расписание для водителей;
- расписание для кондукторов;

- закрепление графика выпуска за ПЕ на каждый день месяца;
- закрепление номера смены за ПЕ на каждый день месяца.

В результате работы данного программного комплекса можно получить:

- базовый наряд ПЕ на маршрут (см. рис. 4.9);
- наряд водителя с учетом внесенных изменений (см. рис. 4.9);
- наряд кондуктора с учетом внесенных изменений (рис. 4.16);

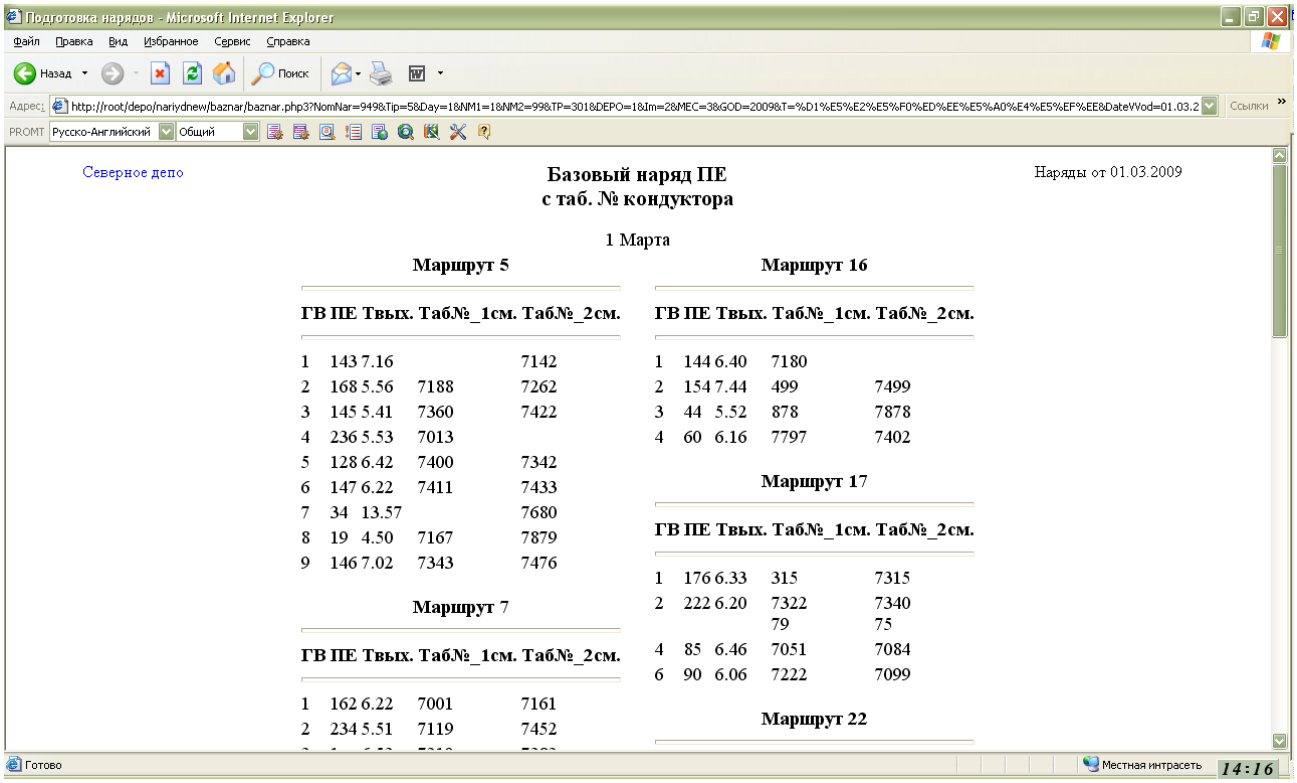


Рис. 4.16. Наряд кондуктора с учетом внесенных изменений

- формы установленных отчетов, доступные в локальной (корпоративной) сети предприятия.

4.4. ПРОГРАММНЫЙ КОМПЛЕКС «ДИСПЕТЧЕР ВЫПУСКА И ДВИЖЕНИЕ ПОДВИЖНОЙ ЕДИНИЦЫ» В СРЕДЕ DELPHI

Программный комплекс «Диспетчер выпуска и движение подвижной единицы» предназначен для автоматизированного контроля над выпуском и движением подвижных единиц, работы водителя и кондуктора на данной ПЕ маршрутизированного городского транспорта.

Возможности программного комплекса:

- подготовка исходных данных по выпуску ПЕ на текущий день с учетом действующего расписания движения маршрутов транспорта;
- оперативное получение информации о закреплении ПЕ за маршрутом движения;

- корректировка базового наряда ПЕ по факту выхода на линию на текущее время;
- корректировка базового наряда водителя по факту выхода на линию на текущее время;
- корректировка базового наряда кондуктора по факту выхода на линию на текущее время;
- корректировка путевого листа водителя;
- автоматизация основных функций диспетчера выпуска ПЕ на линию:

  - изменение табельных номеров водителей и кондукторов;

  - введение замены ПЕ;

  - введение простоев ПЕ;

  - введение резерва водителей и кондукторов;

  - введение (удаление) нового графика маршрута.

Программный комплекс «Диспетчер выпуска и движение подвижной единицы» осуществляет контроль над выпуском на линию запланированного числа технически исправных подвижных единиц, согласно расписанию выхода ПЕ и наряду работы водителей и кондукторов, а также производит своевременную оперативную замену ПЕ как на линии, так и в депо из ПЕ, находящихся в резерве. Диспетчер выпуска следит за заступлением водителей в депо, оформляет путевой лист на ПЕ. Все изменения в путевом листе водителя и кондуктора или в базовом наряде ПЕ оперативно вносятся в данный программный комплекс.

На рис. 4.17 представлена концептуальная модель «Диспетчер выпуска и движение подвижной единицы».

Упрощенная схема базы данных «Диспетчер выпуска и движение подвижной единицы» представлена на рис. 4.18.

Программный комплекс «Диспетчер выпуска и движение подвижной единицы» обеспечивает работу в двух основных режимах: «Оперативный наряд» и «Путевой лист».

В режиме «Оперативный наряд» (см. рис. 4.19) производится:

1. Ввод утренних замен ПЕ, табельных номеров водителей и кондукторов.
2. Изменение времени заступления кондуктора.
3. Ввод и удаление стажера-водителя (см. рис. 4.20).
4. Ввод и удаление нового кондуктора.
5. Ввод и удаление стажера-кондуктора.

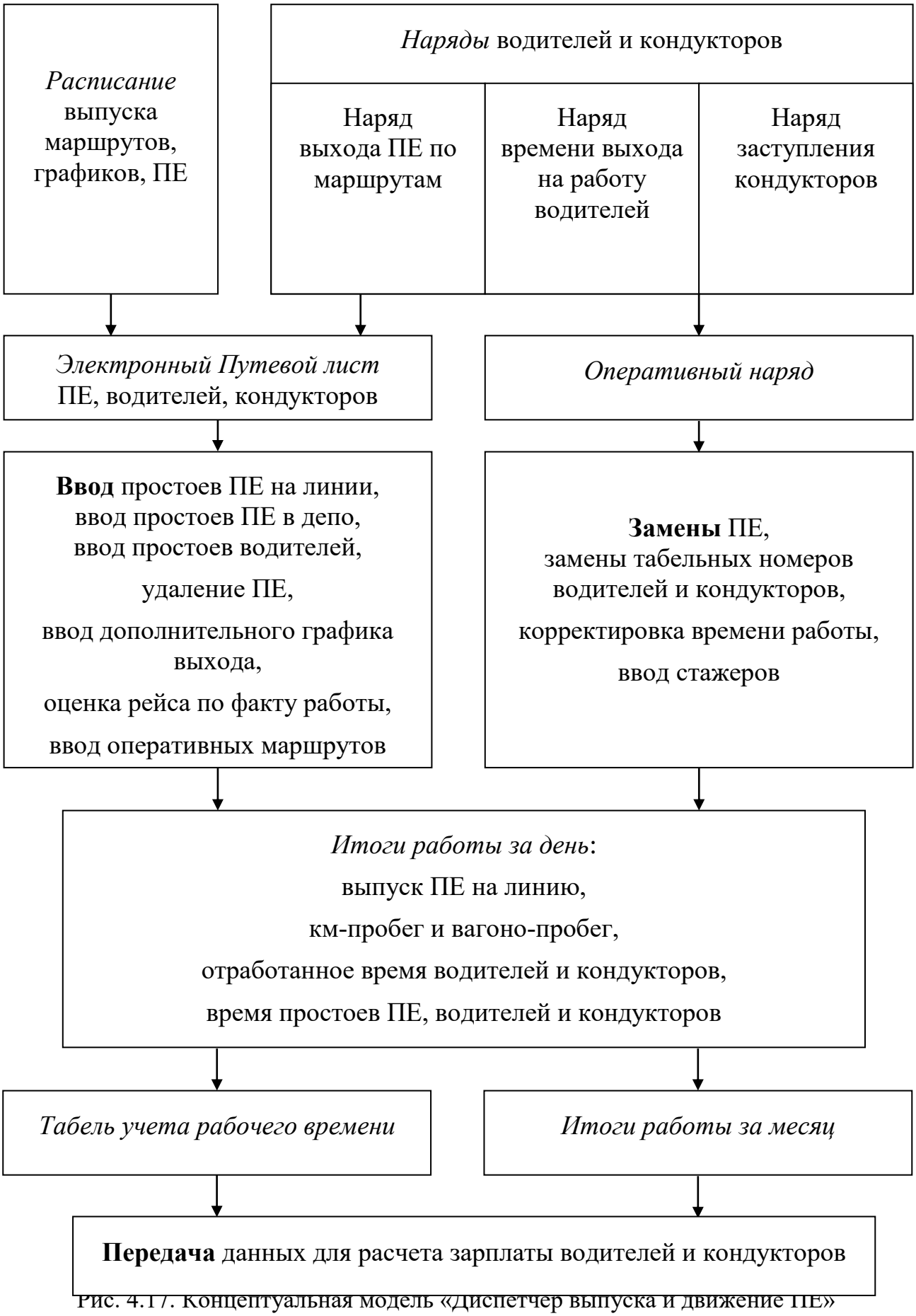
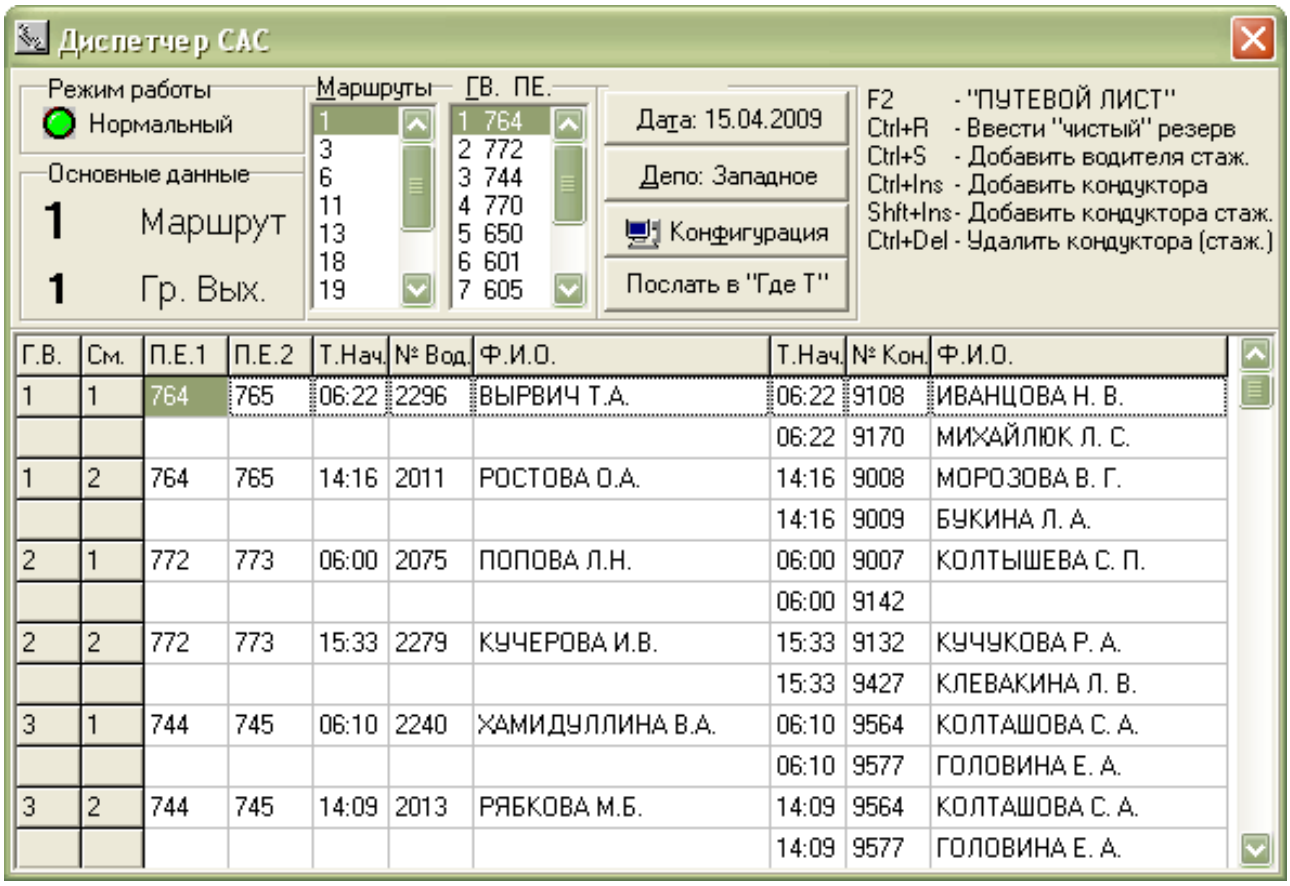


Рис. 4.17. Концептуальная модель «диспетчер выпуска и движение ПЕ»



Г.В.	См.	П.Е.1	П.Е.2	Т.Нач.	№ Вод.	Ф.И.О.	Т.Нач.	№ Кон.	Ф.И.О.
1	1	764	765	06:22	2296	ВЫРВИЧ Т.А.	06:22	9108	ИВАНЦОВА Н. В.
							06:22	9170	МИХАЙЛЮК Л. С.
1	2	764	765	14:16	2011	РОСТОВА О.А.	14:16	9008	МОРОЗОВА В. Г.
							14:16	9009	БУКИНА Л. А.
2	1	772	773	06:00	2075	ПОПОВА Л.Н.	06:00	9007	КОЛТЫШЕВА С. П.
							06:00	9142	
2	2	772	773	15:33	2279	КУЧЕРОВА И.В.	15:33	9132	КУЧУКОВА Р. А.
							15:33	9427	КЛЕВАКИНА Л. В.
3	1	744	745	06:10	2240	ХАМИДУЛЛИНА В.А.	06:10	9564	КОЛТАШОВА С. А.
							06:10	9577	ГОЛОВИНА Е. А.
3	2	744	745	14:09	2013	РЯБКОВА М.Б.	14:09	9564	КОЛТАШОВА С. А.
							14:09	9577	ГОЛОВИНА Е. А.

Рис. 4.19. Оперативный наряд

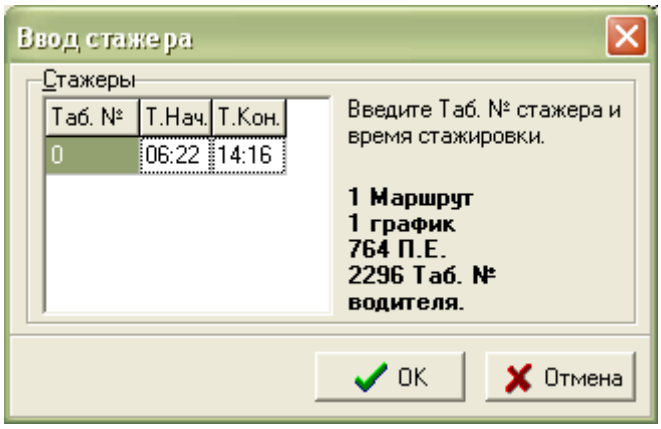


Рис. 4.20. Ввод стажера-водителя

В режиме «Путевой лист» (см. рис. 4.21) производится:

1. Ввод и удаление простоев ПЕ по технической неисправности, с последующей корректировкой любых введенных данных.
2. Ввод и удаление простоев водителей с учетом причин человеческого фактора, с последующей корректировкой любых введенных данных
3. Ввод и удаление простоев кондукторов с учетом причин человеческого фактора, с последующей корректировкой сопутствующих данных.
4. Ввод и удаление замены ПЕ в течение смены (см. рис.4.22).

5. Ввод и удаление замены водителя в течение смены.
6. Изменение времени работы кондуктора.
7. Ввод и удаление нового графика, не отраженного в расписании.
8. Проведение оценки рейса по факту выполнения пробега.
9. Подготовка итоговых выходных данных для использования другими программными комплексами системы (время работы ПЕ, водителей и кондукторов на линии, простои ПЕ и водителей на линии, плановый и фактический пробег ПЕ, плановое и фактическое количество рейсов ПЕ на линии, плановая и фактическая скорость движения).

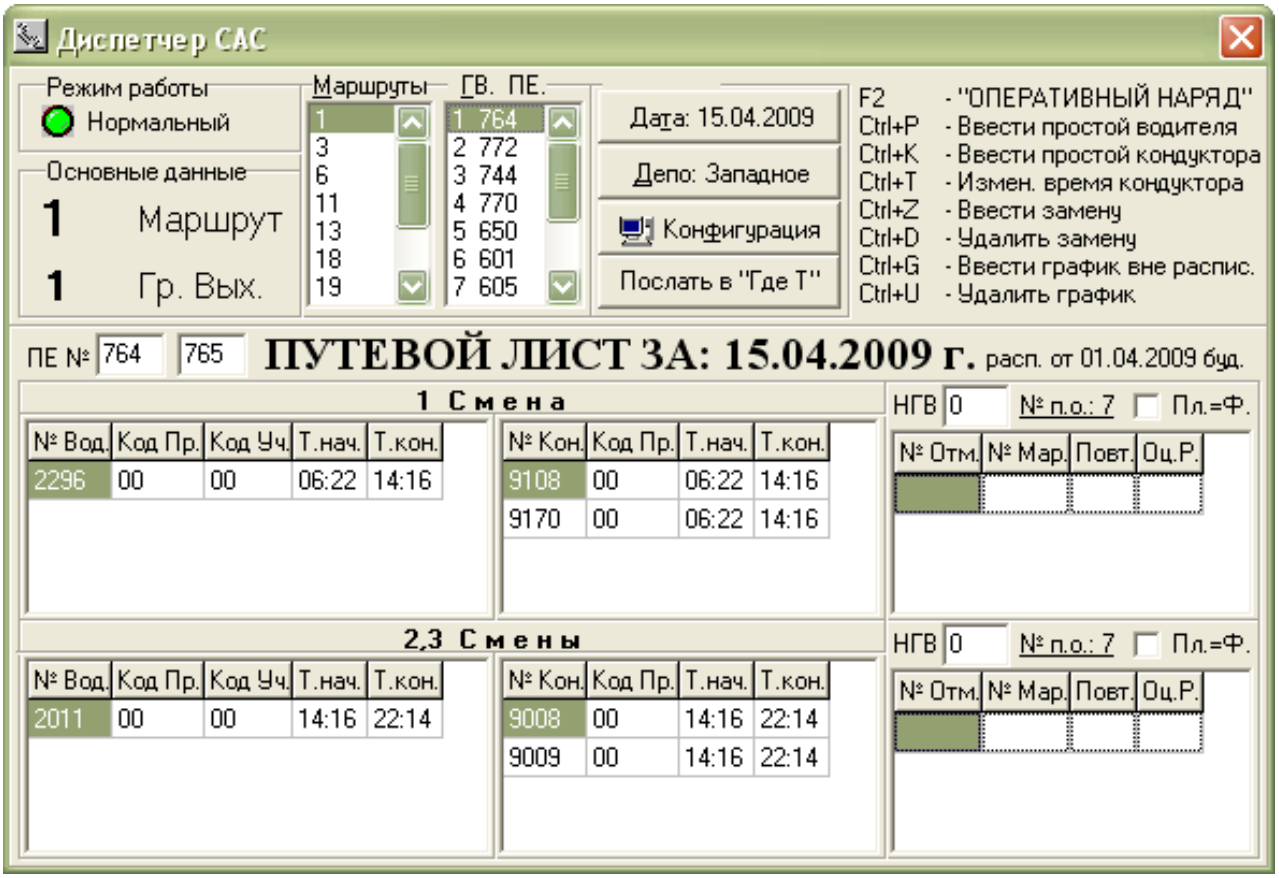


Рис. 4.21. Путевой лист

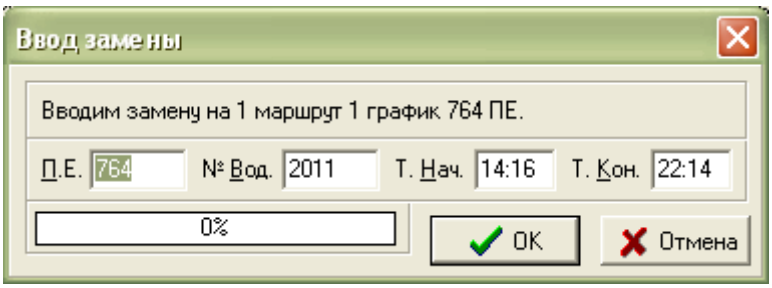


Рис. 4.22. Форма замены ПЕ

Исходные данные программного комплекса «Диспетчер выпуска и движение подвижной единицы»:

- список депо с начальными данными;

- список фамилий водителей и кондукторов;
- список сплоток для трамвая (рис. 4.23);

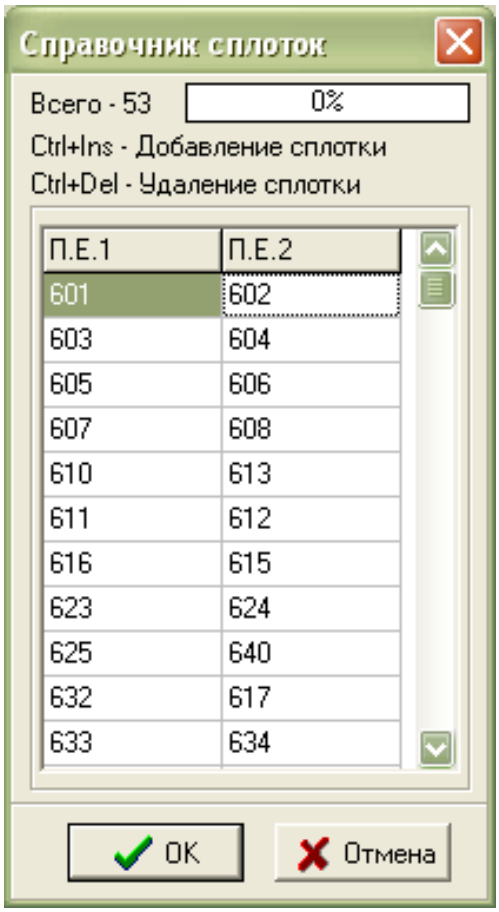


Рис. 4.23. Справочник сплоток

- список кодов простоев (см. рис. 4.24);
- список кодов участков (см. рис. 4.24);
- таблица выпуска по расписанию;
- базовые наряды выпуска по водителям и кондукторам (рассмотрено в разделе «наряды»);
- данные о плановых осмотрах и заходах в депо (см. рис. 4.25);
- список и режимы работы графиков маршрута;
- подготовительно-заключительное время по депо и режиму работы графиков (см. рис. 4.26).

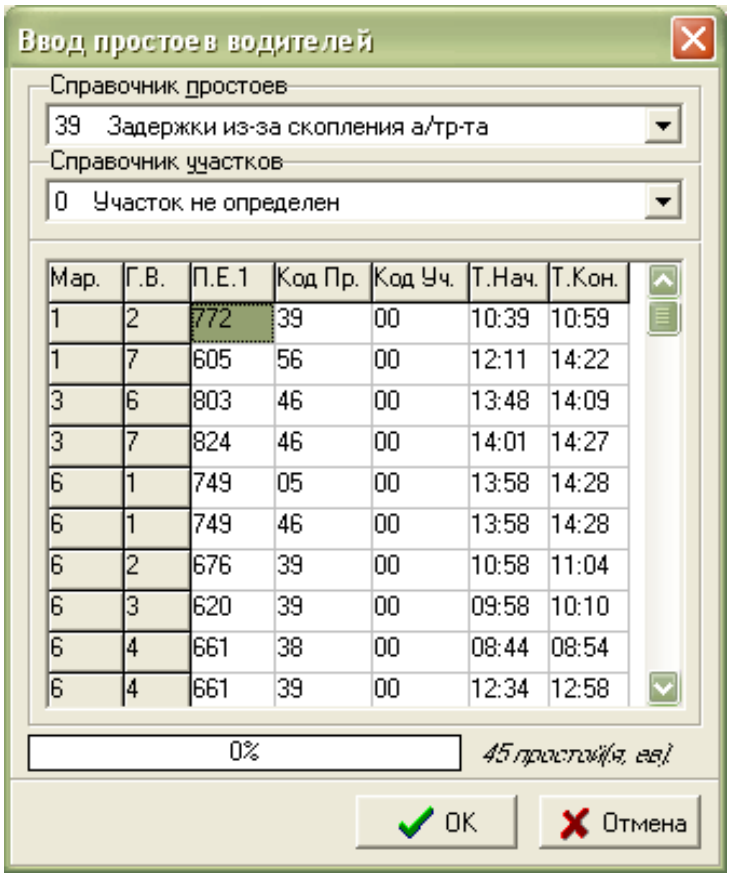


Рис. 4.24. Форма ввода простоев со справочника простоев

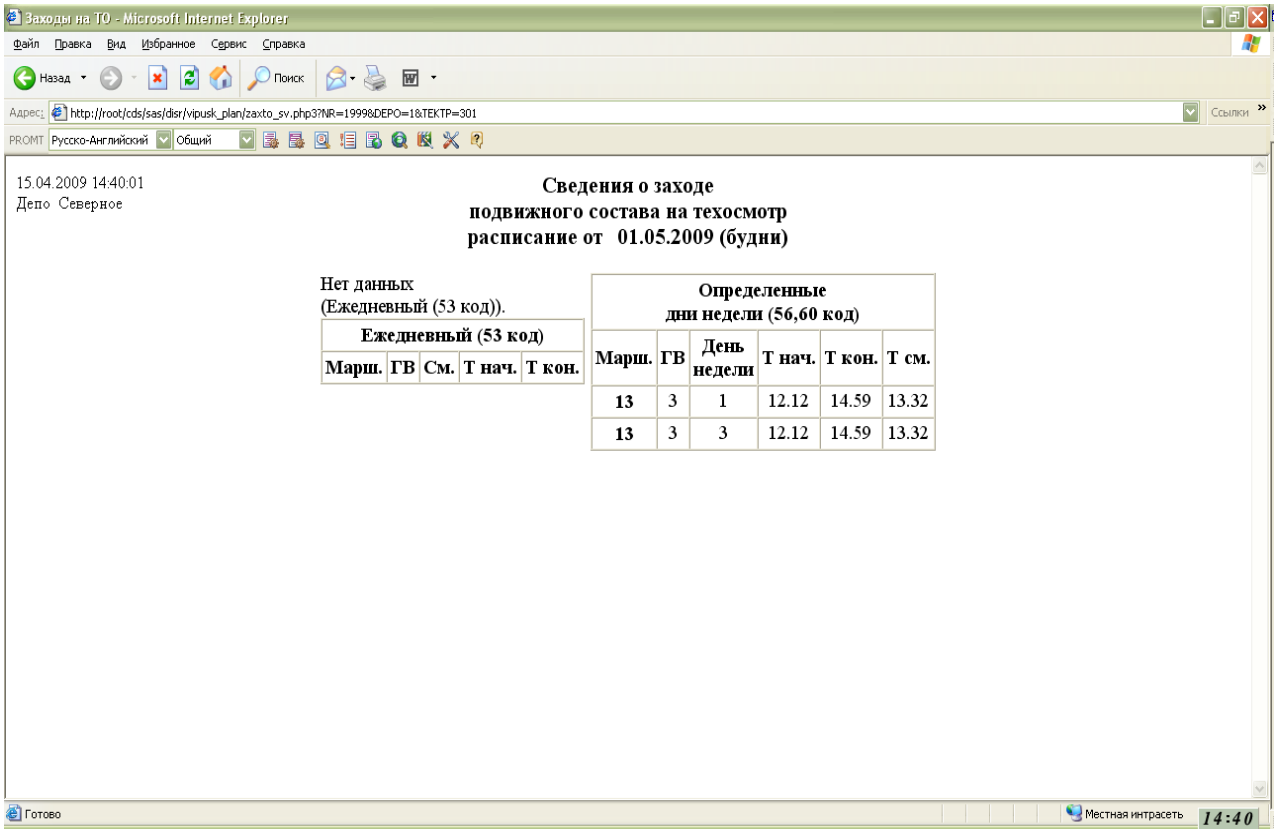
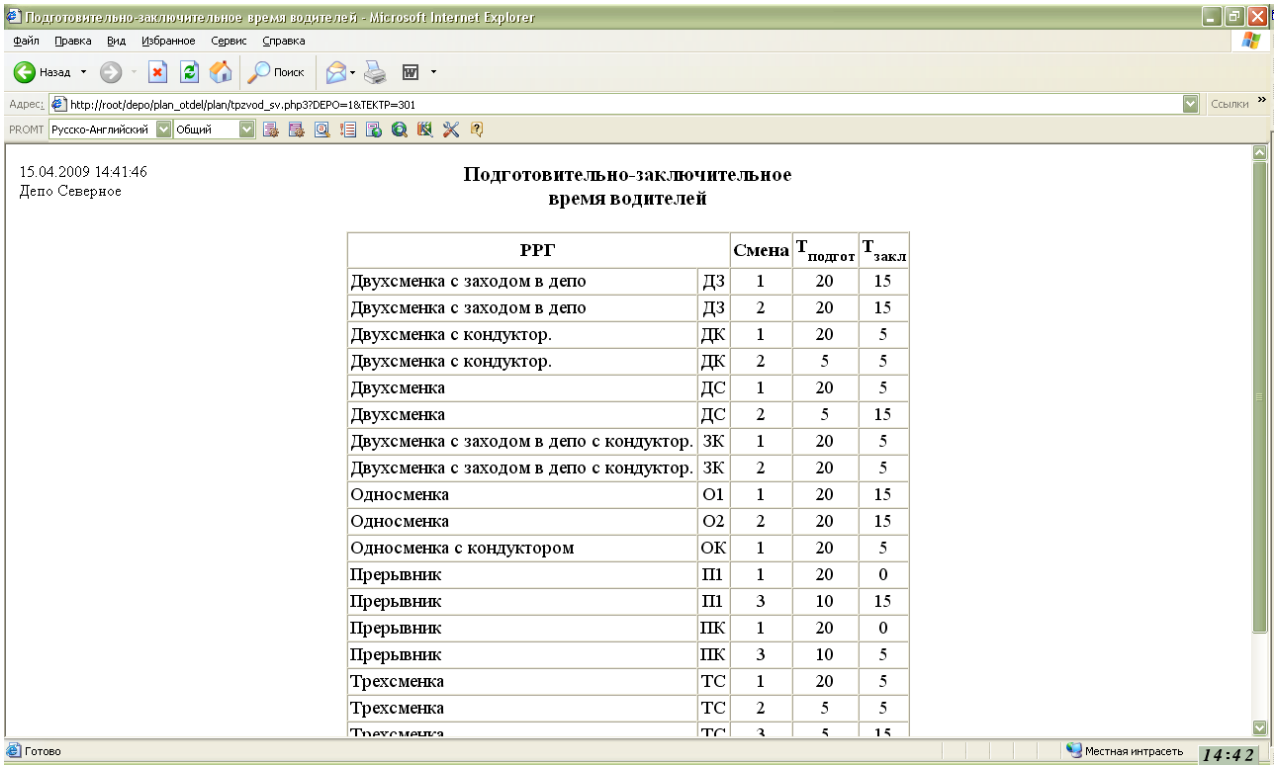


Рис. 4.25. Плановые осмотры и заходы в депо



Подготовительно-заключительное время водителей - Microsoft Internet Explorer

15.04.2009 14:41:46
Депо Северное

Подготовительно-заключительное время водителей

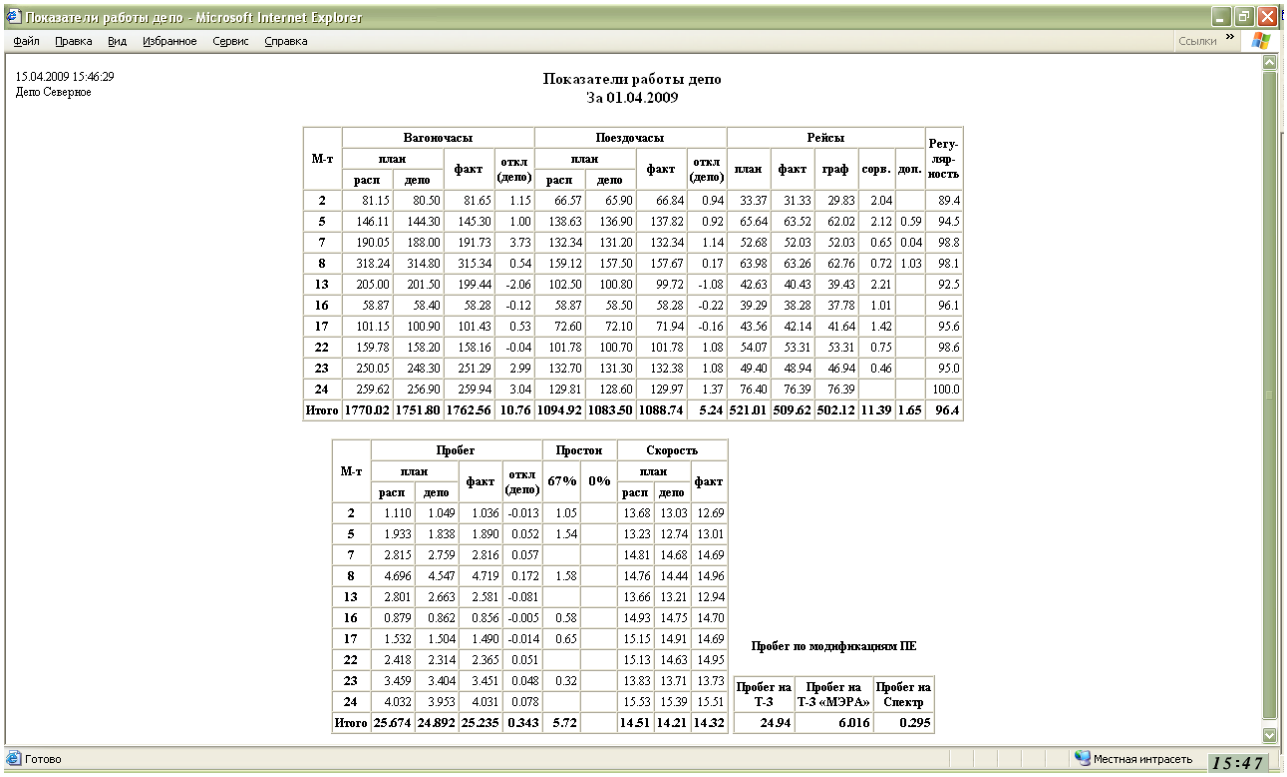
РРГ		Смена	$T_{подгот}$	$T_{закл}$
Двухсменка с заходом в депо	ДЗ	1	20	15
Двухсменка с заходом в депо	ДЗ	2	20	15
Двухсменка с кондуктор.	ДК	1	20	5
Двухсменка с кондуктор.	ДК	2	5	5
Двухсменка	ДС	1	20	5
Двухсменка	ДС	2	5	15
Двухсменка с заходом в депо с кондуктор.	ЗК	1	20	5
Двухсменка с заходом в депо с кондуктор.	ЗК	2	20	5
Односменка	О1	1	20	15
Односменка	О2	2	20	15
Односменка с кондуктором	ОК	1	20	5
Прерывник	П1	1	20	0
Прерывник	П1	3	10	15
Прерывник	ПК	1	20	0
Прерывник	ПК	3	10	5
Трехсменка	ТС	1	20	5
Трехсменка	ТС	2	5	5
Трехсменка	ТС	3	5	15

Готово — Местная интрасеть — 14:42

Рис. 4.26. Подготовительно-заключительное время по депо и режиму работы графиков

В результате работы данного программного комплекса можно получить:

- итоги работы водителей и кондукторов за день;
- формы установленных отчетов (например, рис. 4.27 и 4.28).



Показатели работы депо - Microsoft Internet Explorer

15.04.2009 15:46:29
Депо Северное

Показатели работы депо
За 01.04.2009

М-т	Вагоночасы				Поездочасы				Рейсы					Регулярность
	план		факт	откл (депо)	план		факт	откл (депо)	план	факт	граф	сорв.	доп.	
	расп	депо			расп	депо								
2	81.15	80.50	81.65	1.15	66.57	65.90	66.84	0.94	33.37	31.33	29.83	2.04		89.4
5	146.11	144.30	145.30	1.00	138.63	136.90	137.82	0.92	65.64	63.52	62.02	2.12	0.59	94.5
7	190.05	188.00	191.73	3.73	132.34	131.20	132.34	1.14	52.68	52.03	52.03	0.65	0.04	98.8
8	318.24	314.80	315.34	0.54	159.12	157.50	157.67	0.17	63.98	63.26	62.76	0.72	1.03	98.1
13	205.00	201.50	199.44	-2.06	102.50	100.80	99.72	-1.08	42.63	40.43	39.43	2.21		92.5
16	58.87	58.40	58.28	-0.12	58.87	58.50	58.28	-0.22	39.29	38.28	37.78	1.01		96.1
17	101.15	100.90	101.43	0.53	72.60	72.10	71.94	-0.16	43.56	42.14	41.64	1.42		95.6
22	159.78	158.20	158.16	-0.04	101.78	100.70	101.78	1.08	54.07	53.31	53.31	0.75		98.6
23	250.05	248.30	251.29	2.99	132.70	131.30	132.38	1.08	49.40	48.94	46.94	0.46		95.0
24	259.62	256.90	259.94	3.04	129.81	128.60	129.97	1.37	76.40	76.39	76.39			100.0
Итого	**1770.02**	**1751.80**	**1762.56**	**10.76**	**1094.92**	**1083.50**	**1088.74**	**5.24**	**521.01**	**509.62**	**502.12**	**11.39**	**1.65**	**96.4**

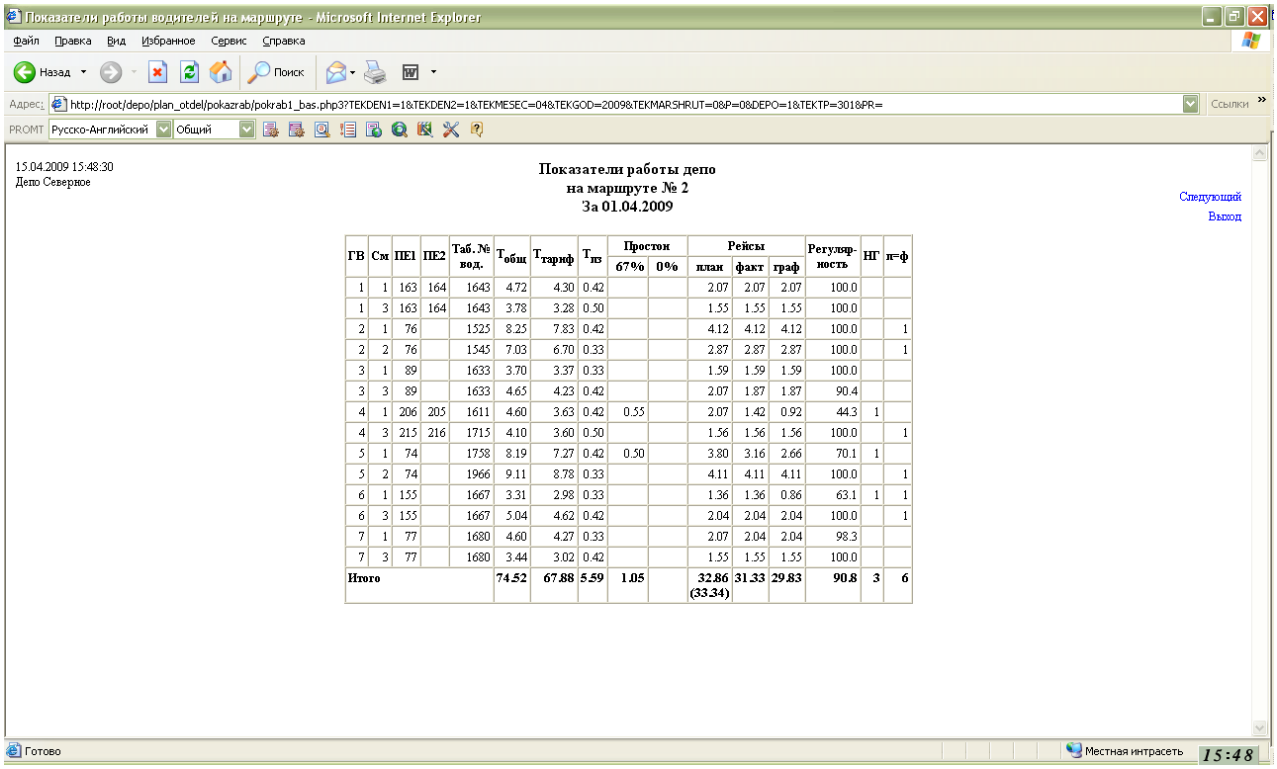
М-т	Пробег				Простои		Скорость		
	план		факт	откл (депо)	67%	0%	план		факт
	расп	депо					расп	депо	
2	1.110	1.049	1.036	-0.013	1.05		13.68	13.03	12.69
5	1.933	1.838	1.890	0.052	1.54		13.23	12.74	13.01
7	2.815	2.759	2.816	0.057			14.81	14.68	14.69
8	4.696	4.547	4.719	0.172	1.58		14.76	14.44	14.96
13	2.801	2.663	2.581	-0.081			13.66	13.21	12.94
16	0.879	0.862	0.856	-0.005	0.58		14.93	14.75	14.70
17	1.532	1.504	1.490	-0.014	0.65		15.15	14.91	14.69
22	2.418	2.314	2.365	0.051			15.13	14.63	14.95
23	3.459	3.404	3.451	0.048	0.32		13.83	13.71	13.73
24	4.032	3.953	4.031	0.078			15.53	15.39	15.51
Итого	**25.674**	**24.892**	**25.235**	**0.343**	**5.72**		**14.51**	**14.21**	**14.32**

Пробег по модификациям ПЕ

Пробег на Т-3	Пробег на Т-3 «МЭРА»	Пробег на Спектр
24.94	**6.016**	**0.295**

Готово — Местная интрасеть — 15:47

Рис. 4.27. Показатели работы депо за день



ГВ	См	ПЕ1	ПЕ2	Таб. № вод.	$T_{общ}$	$T_{тариф}$	$T_{нз}$	Простои		Рейсы			Регуляр-ность	НГ	н=ф
								67%	0%	план	факт	граф			
1	1	163	164	1643	4.72	4.30	0.42			2.07	2.07	2.07	100.0		
1	3	163	164	1643	3.78	3.28	0.50			1.55	1.55	1.55	100.0		
2	1	76		1525	8.25	7.83	0.42			4.12	4.12	4.12	100.0		1
2	2	76		1545	7.03	6.70	0.33			2.87	2.87	2.87	100.0		1
3	1	89		1633	3.70	3.37	0.33			1.59	1.59	1.59	100.0		
3	3	89		1633	4.65	4.23	0.42			2.07	1.87	1.87	90.4		
4	1	206	205	1611	4.60	3.63	0.42	0.55		2.07	1.42	0.92	44.3	1	
4	3	215	216	1715	4.10	3.60	0.50			1.56	1.56	1.56	100.0		1
5	1	74		1758	8.19	7.27	0.42	0.50		3.80	3.16	2.66	70.1	1	
5	2	74		1966	9.11	8.78	0.33			4.11	4.11	4.11	100.0		1
6	1	155		1667	3.31	2.98	0.33			1.36	1.36	0.86	63.1	1	1
6	3	155		1667	5.04	4.62	0.42			2.04	2.04	2.04	100.0		1
7	1	77		1680	4.60	4.27	0.33			2.07	2.04	2.04	98.3		
7	3	77		1680	3.44	3.02	0.42			1.55	1.55	1.55	100.0		
Итого					**74.52**	**67.88**	**5.59**	**1.05**		**32.86 (33.34)**	**31.33**	**29.83**	**90.8**	**3**	**6**

Рис. 4.28. Показатели работы маршрута за день

4.5. ПРОГРАММНЫЙ КОМПЛЕКС «ТАБЕЛЬ УЧЕТА РАБОЧЕГО ВРЕМЕНИ ВОДИТЕЛЕЙ И КОНДУКТОРОВ» В СРЕДЕ DELPHI

Программный комплекс «Табель учета рабочего времени водителей и кондукторов» позволяет вести электронный учет рабочего времени водителей и кондукторов, отработанные ими сверхустановленной нормы (снятие с выходного, сверхурочные) для текущей и последующей оплаты. Методика автоматизированного расчета рабочего времени работникам, которым установлен суммированный учет рабочего времени, утверждена на предприятии ЕМУП ТТУ с учетом Трудового кодекса России.

Программный комплекс «Табель учета рабочего времени водителей и кондукторов» позволяет осуществлять:

- получение отчетных данных по каждому водителю и кондуктору;
- корректировку нормы рабочего времени на основании причин отсутствия;
- расчет рабочего времени в случае снятия с выходных дней;
- расчет сверхурочных часов рабочего времени.

На рис. 4.29 представлена концептуальная модель «Табель учета рабочего времени водителей и кондукторов».

Упрощенная схема базы данных «Табель учета рабочего времени водителей и кондукторов» совместно с «Подготовкой нарядов водителей и кондукторов» была представлена на рис. 4.11.

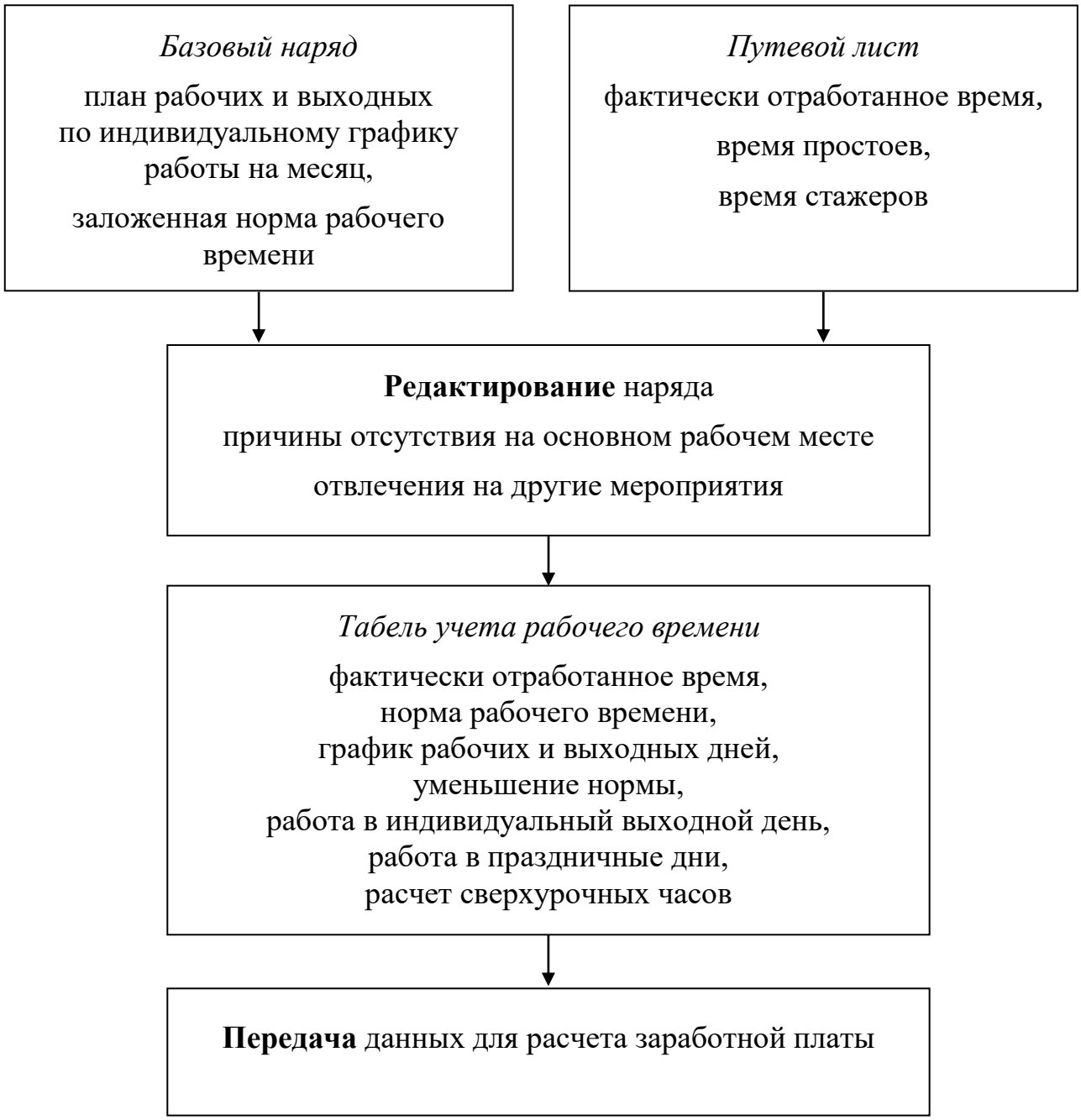


Рис. 4.29. Концептуальная модель «Табель учета рабочего времени водителей и кондукторов»

Исходные данные программного комплекса «Табель учета рабочего времени водителей и кондукторов»:

- плановый индивидуальный наряд-закрепление по режимам работы водителей и кондукторов на месяц (квартал), предварительно рассчитанный и откорректированный в депо, с учетом текущей работы и нормативно-справочных данных;
- индивидуальный производственный календарь на год с учетом праздничных дней и переносов рабочих и выходных дней;

- отработанное время (общее время, время по тарифам, праздничное время, время простоев);
- причины отсутствия водителей и кондукторов на основном рабочем месте;
- прочие явки на работу.

Общее время, время по тарифам, праздничное время, время простоев – это расчетные величины из программного комплекса «Путевой лист водителя и кондуктора». Причины отсутствий вводятся нарядчиком на основании соответствующих предоставленных документов. Прочие явки – перевод на другую работу для выполнения производственных и иных работ, документально подтвержденных соответствующими документами.

В результате работы данного программного комплекса можно получить:

- табель учета рабочего времени водителей и кондукторов (рис. 4.30);

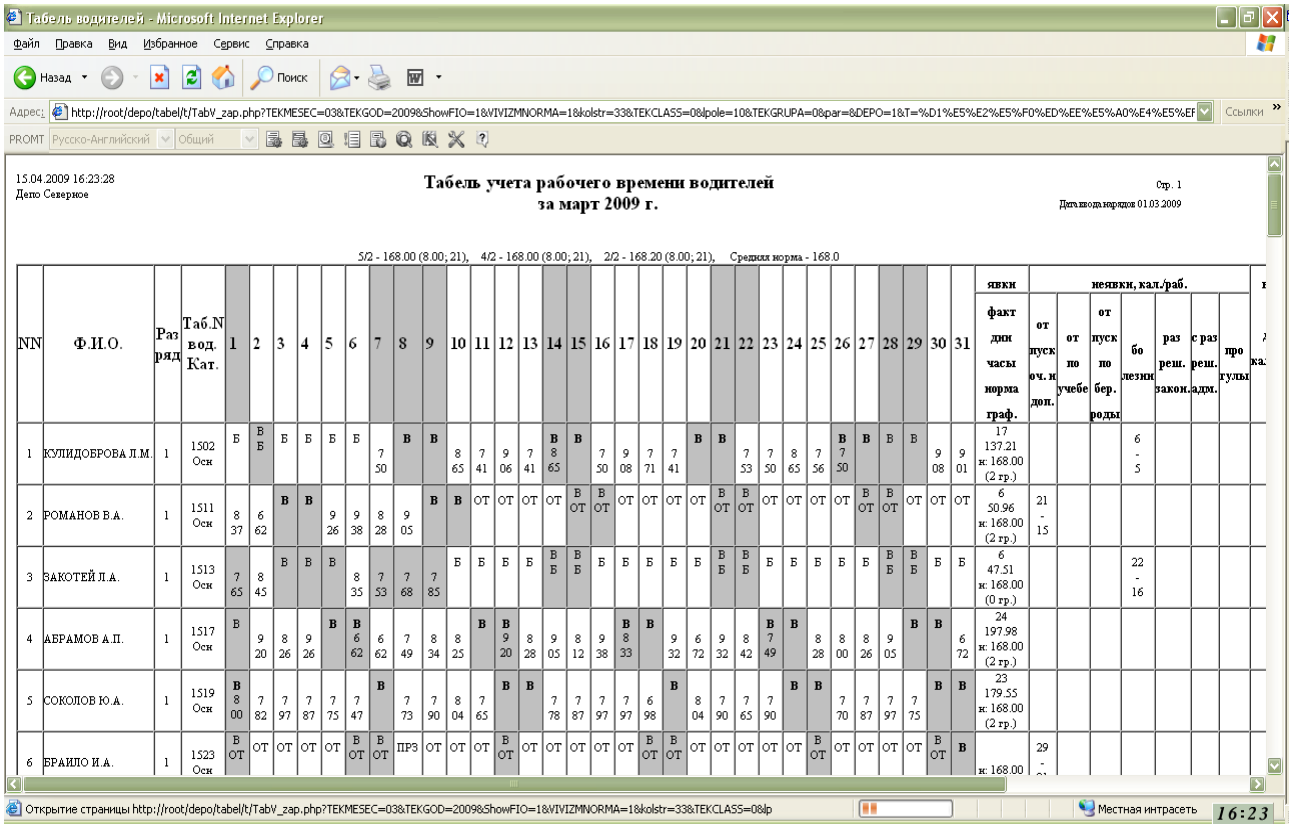
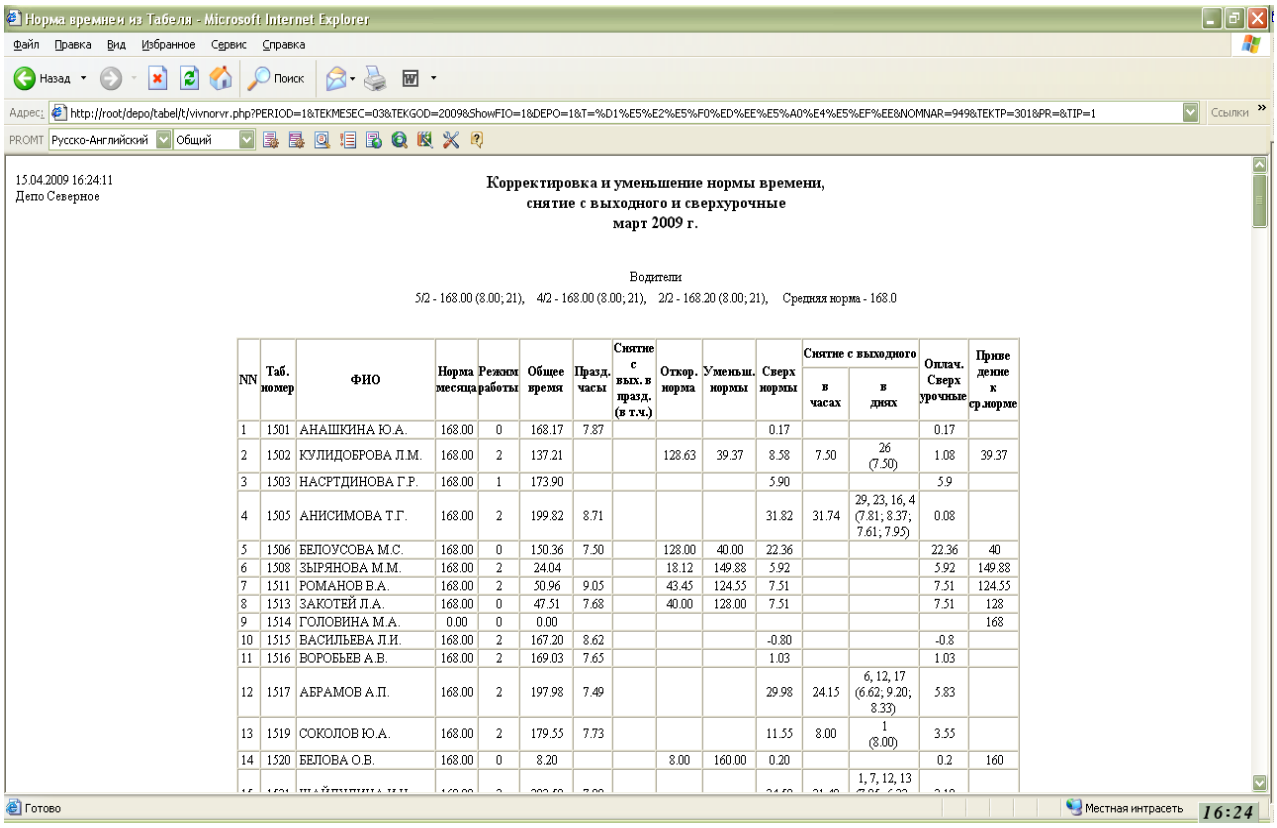


Рис. 4.30. Табель учета рабочего времени водителей и кондукторов

- данные для расчета заработной платы водителей и кондукторов (см. рис. 4.31, 4.32).



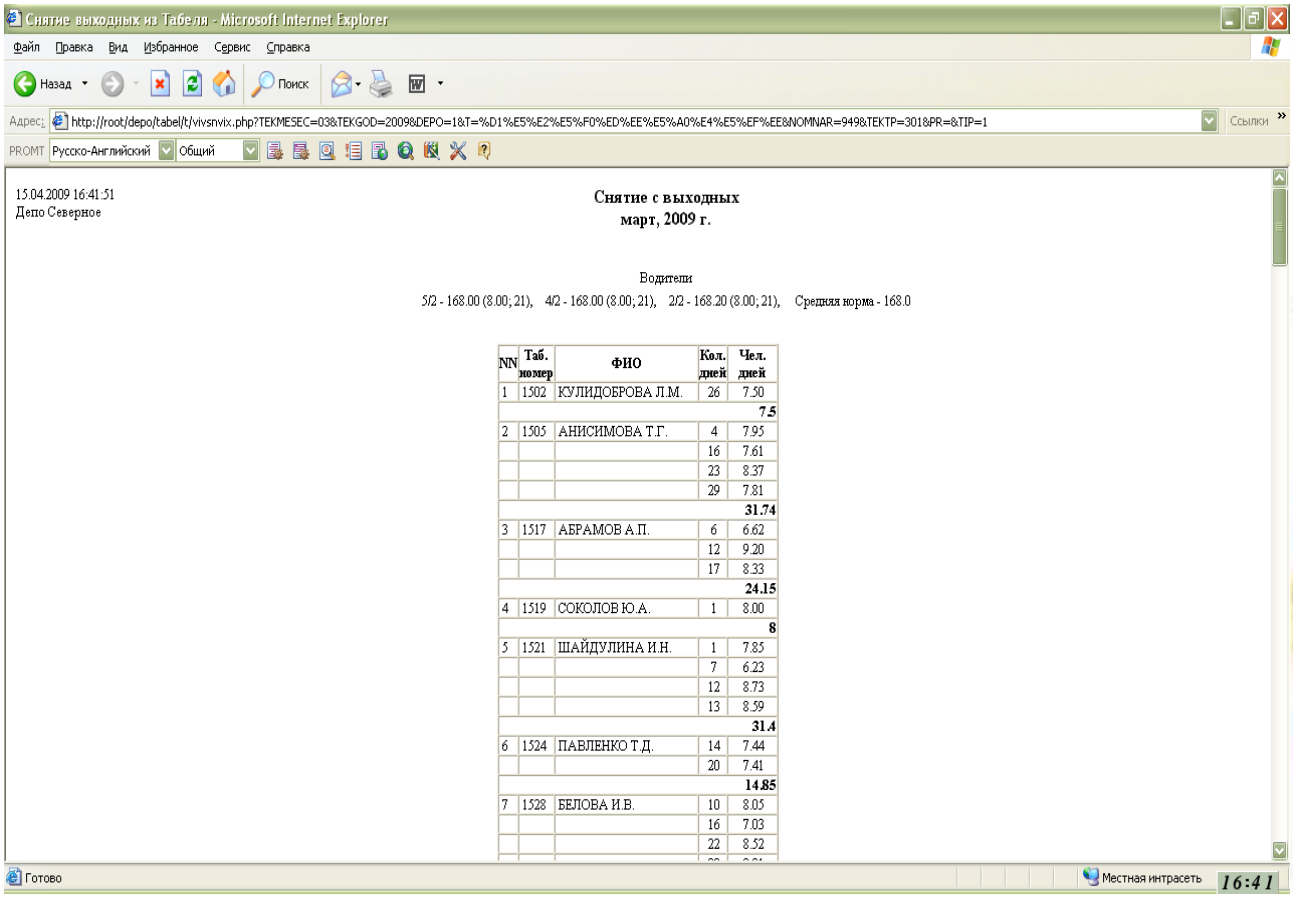
**Корректировка и уменьшение нормы времени,
снятие с выходного и сверхурочные
март 2009 г.**

Водители

5/2 - 168.00 (8.00; 21), 4/2 - 168.00 (8.00; 21), 2/2 - 168.20 (8.00; 21), Средняя норма - 168.0

NN	Таб. номер	ФИО	Норма месяца	Режим работы	Общее время	Празд. часы	Снятие с вых. в празд. (в т.ч.)	Откор. норма	Уменьш. нормы	Сверх нормы	Снятие с выходного		Оплач. Сверх урочные	Приведение к ср.норме
											в часах	в днях		
1	1501	АНАШКИНА Ю.А.	168.00	0	168.17	7.87				0.17			0.17	
2	1502	КУЛИДОБРОВА Л.М.	168.00	2	137.21			128.63	39.37	8.58	7.50	26 (7.50)	1.08	39.37
3	1503	НАСРТДИНОВА Г.Р.	168.00	1	173.90					5.90			5.9	
4	1505	АНИСИМОВА Т.Г.	168.00	2	199.82	8.71				31.82	31.74	29, 23, 16, 4 (7.81; 8.37; 7.61; 7.95)	0.08	
5	1506	БЕЛОУСОВА М.С.	168.00	0	150.36	7.50		128.00	40.00	22.36			22.36	40
6	1508	ЗЫРЯНОВА М.М.	168.00	2	24.04			18.12	149.88	5.92			5.92	149.88
7	1511	РОМАНОВ В.А.	168.00	2	50.96	9.05		43.45	124.55	7.51			7.51	124.55
8	1513	ЗАКОТЕЙ Л.А.	168.00	0	47.51	7.68		40.00	128.00	7.51			7.51	128
9	1514	ГОЛОВИНА М.А.	0.00	0	0.00									168
10	1515	ВАСИЛЬЕВА Л.И.	168.00	2	167.20	8.62				-0.80			-0.8	
11	1516	ВОРОБЬЕВ А.В.	168.00	2	169.03	7.65				1.03			1.03	
12	1517	АБРАМОВ А.П.	168.00	2	197.98	7.49				29.98	24.15	6, 12, 17 (6.62; 9.20; 8.33)	5.83	
13	1519	СОКОЛОВ Ю.А.	168.00	2	179.55	7.73				11.55	8.00	1 (8.00)	3.55	
14	1520	БЕЛОВА О.В.	168.00	0	8.20			8.00	160.00	0.20			0.2	160
												1, 7, 12, 13		

Рис. 4.31. Отчетный файл о работе водителей и кондукторов



**Снятие с выходных
март, 2009 г.**

Водители

5/2 - 168.00 (8.00; 21), 4/2 - 168.00 (8.00; 21), 2/2 - 168.20 (8.00; 21), Средняя норма - 168.0

NN	Таб. номер	ФИО	Кол. дней	Чел. дней
1	1502	КУЛИДОБРОВА Л.М.	26	7.50
				7.5
2	1505	АНИСИМОВА Т.Г.	4	7.95
			16	7.61
			23	8.37
			29	7.81
				31.74
3	1517	АБРАМОВ А.П.	6	6.62
			12	9.20
			17	8.33
				24.15
4	1519	СОКОЛОВ Ю.А.	1	8.00
				8
5	1521	ШАЙДУЛИНА И.Н.	1	7.85
			7	6.23
			12	8.73
			13	8.59
				31.4
6	1524	ПАВЛЕНКО Т.Д.	14	7.44
			20	7.41
				14.85
7	1528	БЕЛОВА И.В.	10	8.05
			16	7.03
			22	8.52

Рис. 4.32. Файл о снятии с выходных водителей и кондукторов

4.6. ПРОГРАММНЫЙ КОМПЛЕКС «ОБРАБОТКА ПУТЕВОГО ЛИСТА АВТОТРАНСПОРТНОЙ СЛУЖБЫ» В СРЕДЕ DELPHI

Программный комплекс «Путевой лист автотранспортной службы» предназначен для контроля движением подвижных единиц (автомашин), за выпуском их из гаража и въездом в гараж, а также расходом топлива подвижных единиц.

На рис. 4.33 представлена концептуальная модель «Путевой лист автотранспортной службы».

Упрощенная схема базы данных «Путевой лист автотранспортной службы» представлена на рис. 4.34.

Исходные данные программного комплекса «Путевой лист автотранспортной службы»:

- номер машины;
- фамилия водителя;
- дата работы;
- время начала и окончания работы автомашины;
- вид транспорта (легковая, грузовая, самосвал, компрессор и т.д.);
- марка (ВАЗ, ГАЗ и т.д.);
- подразделение приписки (к какому подразделения предприятия она относится);
- тип топлива (А-76, АИ-92, дизельное топливо);
- номер заправочной карты топлива с нефтезаправочной станции.

Автоматически с последнего путевого листа в новый переносятся начальный и конечный остаток бензина, начальное и конечное показание спидометра автомашины.

На любом этапе ввода и изменения данных путевого листа производится оценка рейса по следующим показателям:

- пробегу автомашины;
- плановому расходу горючего;
- фактическому расходу горючего;
- экономии или перерасходу горючего;
- рабочему времени работы водителя;
- ночному (с 22 ч. 30 мин. по 6 ч. 00 мин.) времени работы водителя;
- работе в другом подразделении;
- работа в командировке;

- работа без спидометра.

Рис. 4.33. Концептуальная модель «Путевой лист автотранспортной службы»

Эти показатели определяются на основе следующих параметров: пробег, конечные и начальные показания спидометра, фактический расход бензина, показания спидометра при выезде из гаража, количество литров, которыми водитель заправил машину на заправочной станции, показания спидометра при заезде в гараж, количество бензина, слитого с машины, количество бензина, слитого на машину, количество бензина, слитого с машины на хозяйственные нужды, плановый расход горючего, фактический общий пробег автомашины, пробег автомашины, при выполнении дополнительной работы (буксировка, пробег с грузом, пробег за городом), норма основного пробега (норма – это расход горючего на 100 км пробега в данном сезоне, сезон – время года, в зависимости от климатических условий (зима, лето), время начала и конца смены, время обеда (0,8 часа для всех машин кроме аварийных, у аварийных машин – 0,75 часа), признак аварийных машин, признак нового двигателя (пробег двигателя не превышает 2 тыс. км).

Возможна обработка дополнительной работы, выполняемой на данном автомобиле из справочника «Нормы расхода горючего на дополнительную работу», например: пробег за городом, пробег с грузом, буксировка, прогрев двигателя и прочее.

Для автоматизации работы оператора используются следующие справочники:

- справочник машин;
- справочник водителей;
- справочник марки машин;
- справочник нормы расхода топлива;
- справочник видов техники;
- справочник хозяйственных работ;
- дополнительные работы;
- показатели расчета нормы дополнительной работы;
- справочник подразделений;
- календарь;
- справочник сезонов.

В результате работы данного программного комплекса можно получить:

- путевой лист для каждой автомашины, в которой указывается следующая информация:
  - общий пробег;
  - пробег для выполнения дополнительной работы;
  - получение горючего;

– расход топлива по норме, фактический;

– экономия или перерасход топлива;

– остатки горючего на начало текущего и следующего месяца;

– машино-часы в движении.

- отчетные документы, доступные на *web*-сервере.

Кроме того, при наличии данных о заправке водителей на автозаправочных станциях можно провести сравнительный анализ данных путевого листа автомашины и данных с нефтезаправочных станций о заправке бензина на каждую автомашину.

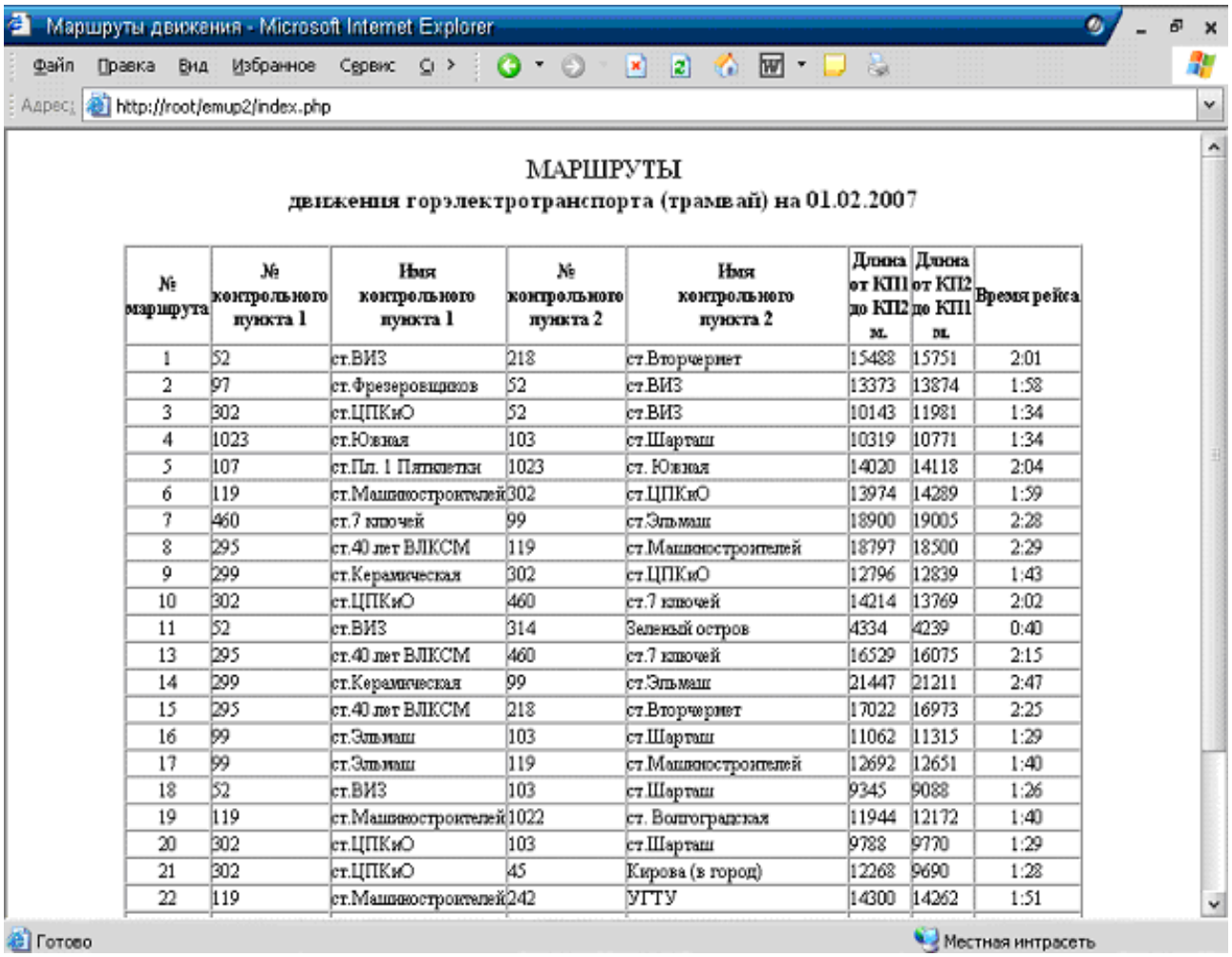
4.7. ПРОГРАММНЫЙ КОМПЛЕКС «ТРАНСПОРТ ГОРОДА ЕКАТЕРИНБУРГА» В СРЕДЕ PHP

В рамках городской целевой программы «Электронный Екатеринбург», утвержденной Екатеринбургской городской Думой, внедряется программный комплекс «Транспорт города Екатеринбурга» в виде информационного сайта. Организатором является администрация города Екатеринбурга, в лице Комитета по промышленности, науке, связи и информационным технологиям и Комитета по транспорту и организации дорожного движения. В настоящее время выполнена интеграция сайта с системой АСУ ЕМУП ТТУ. Информационный сайт находится в открытом доступе и предназначен для всех пользователей Интернет-сети.

В программном комплексе «Транспорт города Екатеринбурга» формируются в формате *html* электронные документы по утвержденной в ЕМУП ТТУ структуре, содержащие следующую информацию:

- маршруты движения трамваев и троллейбусов в г. Екатеринбург (см. рис. 4.35);
- конфигурация маршрутов трамваев и троллейбусов в г. Екатеринбург (см. рис. 4.36);
- расписание движения трамваев и троллейбусов в г. Екатеринбург для населения;
- выполнение плана рейсов трамваев и троллейбусов;
- выпуск подвижного состава трамваев и троллейбусов по времени (см. рис. 4.37);
- справка о сходе трамваев и троллейбусов;
- интервал движения по маршрутам по периодам времени трамваев и троллейбусов (см. рис. 4.38);
- сведения о движении поездов на линии по маршрутам трамваев и троллейбусов;

- время прохождения станции трамваев и троллейбусов;
- расписание (режим) работы трамваев и троллейбусов (см. рис. 4.39);
- задержки движения на линии и в депо по техническим неисправностям трамваев и троллейбусов (см. рис. 4.40);
- показатели работы МУП ЕТТУ (см. рис. 4.41).



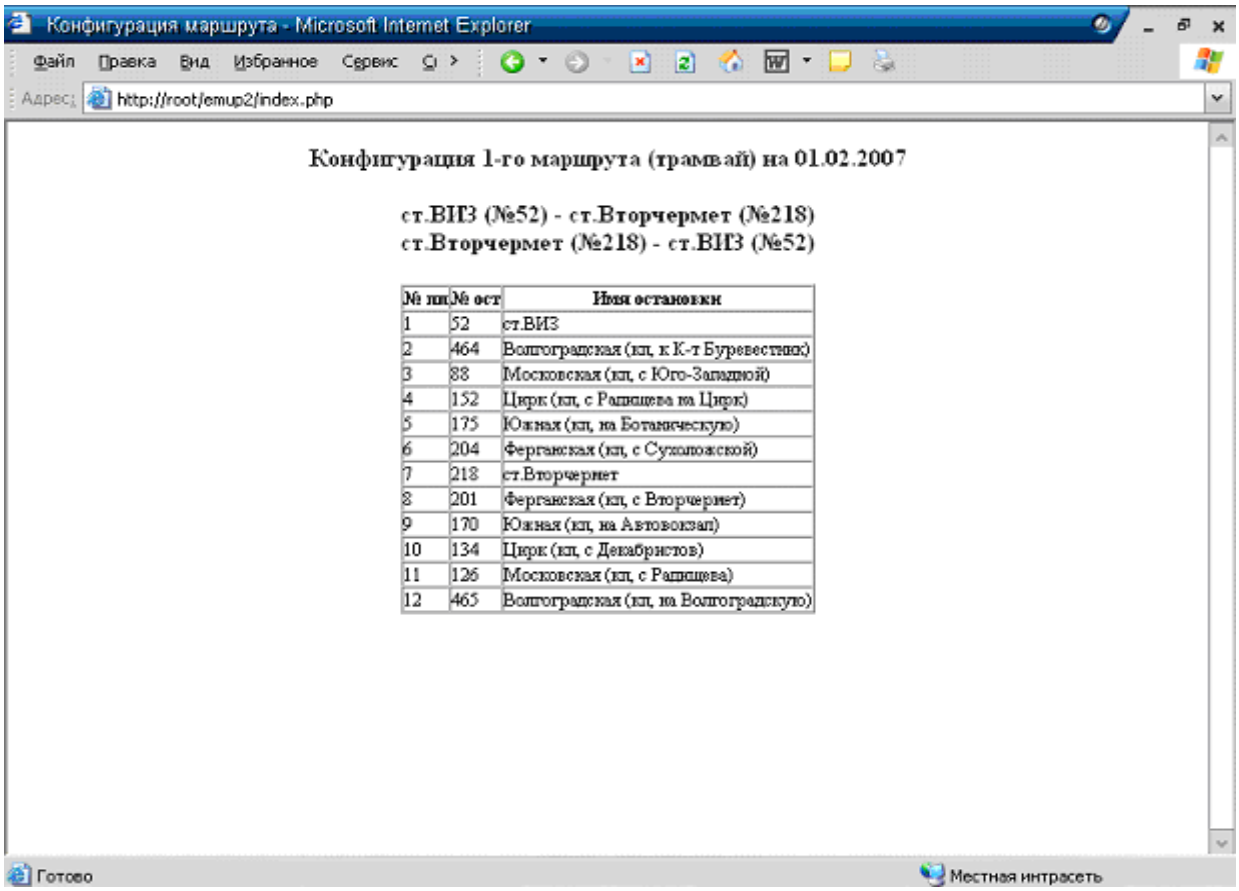
МАРШРУТЫ
движения горэлектротранспорта (трамвай) на 01.02.2007

№ маршрута	№ контрольного пункта 1	Имя контрольного пункта 1	№ контрольного пункта 2	Имя контрольного пункта 2	Длина от КП1 до КП2 м.	Длина от КП2 до КП1 м.	Время рейса
1	52	ст.ВИЗ	218	ст.Вторчермет	15488	15751	2:01
2	97	ст.Фрезеровщиков	52	ст.ВИЗ	13373	13874	1:58
3	302	ст.ЦПКиО	52	ст.ВИЗ	10143	11981	1:34
4	1023	ст.Южная	103	ст.Шарташ	10319	10771	1:34
5	107	ст.Пл. 1 Пятилетки	1023	ст. Южная	14020	14118	2:04
6	119	ст.Машиностроителей	302	ст.ЦПКиО	13974	14289	1:59
7	460	ст.7 ключей	99	ст.Эльмаш	18900	19005	2:28
8	295	ст.40 лет ВЛКСМ	119	ст.Машиностроителей	18797	18500	2:29
9	299	ст.Керамическая	302	ст.ЦПКиО	12796	12839	1:43
10	302	ст.ЦПКиО	460	ст.7 ключей	14214	13769	2:02
11	52	ст.ВИЗ	314	Зеленый остров	4334	4239	0:40
13	295	ст.40 лет ВЛКСМ	460	ст.7 ключей	16529	16075	2:15
14	299	ст.Керамическая	99	ст.Эльмаш	21447	21211	2:47
15	295	ст.40 лет ВЛКСМ	218	ст.Вторчермет	17022	16973	2:25
16	99	ст.Эльмаш	103	ст.Шарташ	11062	11315	1:29
17	99	ст.Эльмаш	119	ст.Машиностроителей	12692	12651	1:40
18	52	ст.ВИЗ	103	ст.Шарташ	9345	9088	1:26
19	119	ст.Машиностроителей	1022	ст. Волгоградская	11944	12172	1:40
20	302	ст.ЦПКиО	103	ст.Шарташ	9788	9770	1:29
21	302	ст.ЦПКиО	45	Кирова (в город)	12268	9690	1:28
22	119	ст.Машиностроителей	242	УГТУ	14300	14262	1:51

Рис. 4.35. Маршруты движения трамваев и троллейбусов в г. Екатеринбург

Информационный сайт «Транспорт города Екатеринбурга» позволяет:

- обеспечить информационное обеспечение населения о маршрутах трамваев и троллейбусов;
- получить информацию о маршрутах следования любого выбранного маршрута, т.е. описание порядка следования всех остановок маршрута;
- получить сведения об интервале движения маршрутов по конечным станциям по периодам времени;
- получить расписание движения трамваев и троллейбусов по выбранной пользователем станции или остановки города Екатеринбурга.

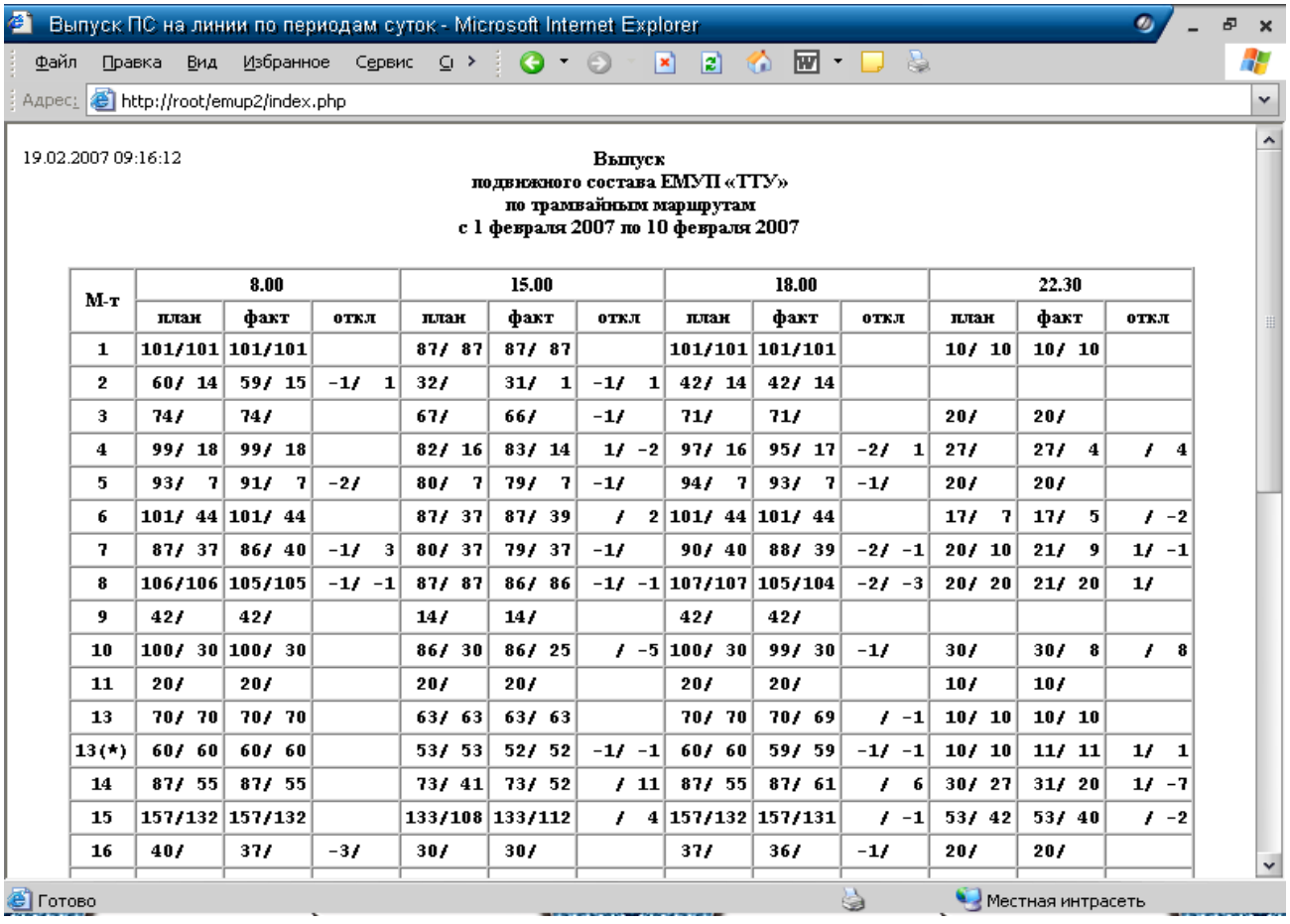


Конфигурация 1-го маршрута (трамвай) на 01.02.2007

ст.ВИЗ (№52) - ст.Вторчермет (№218)
ст.Вторчермет (№218) - ст.ВИЗ (№52)

№ пп	№ ост	Имя остановки
1	52	ст.ВИЗ
2	464	Волгоградская (кп, к К-т Буревестник)
3	88	Московская (кп, с Юго-Западной)
4	152	Цирк (кп, с Радищева на Цирк)
5	175	Южная (кп, на Ботаническую)
6	204	Ферганская (кп, с Сухоложской)
7	218	ст.Вторчермет
8	201	Ферганская (кп, с Вторчермет)
9	170	Южная (кп, на Автовокзал)
10	134	Цирк (кп, с Декабристов)
11	126	Московская (кп, с Радищева)
12	465	Волгоградская (кп, на Волгоградскую)

Рис. 4.36. Конфигурация маршрутов трамваев и троллейбусов в г. Екатеринбург

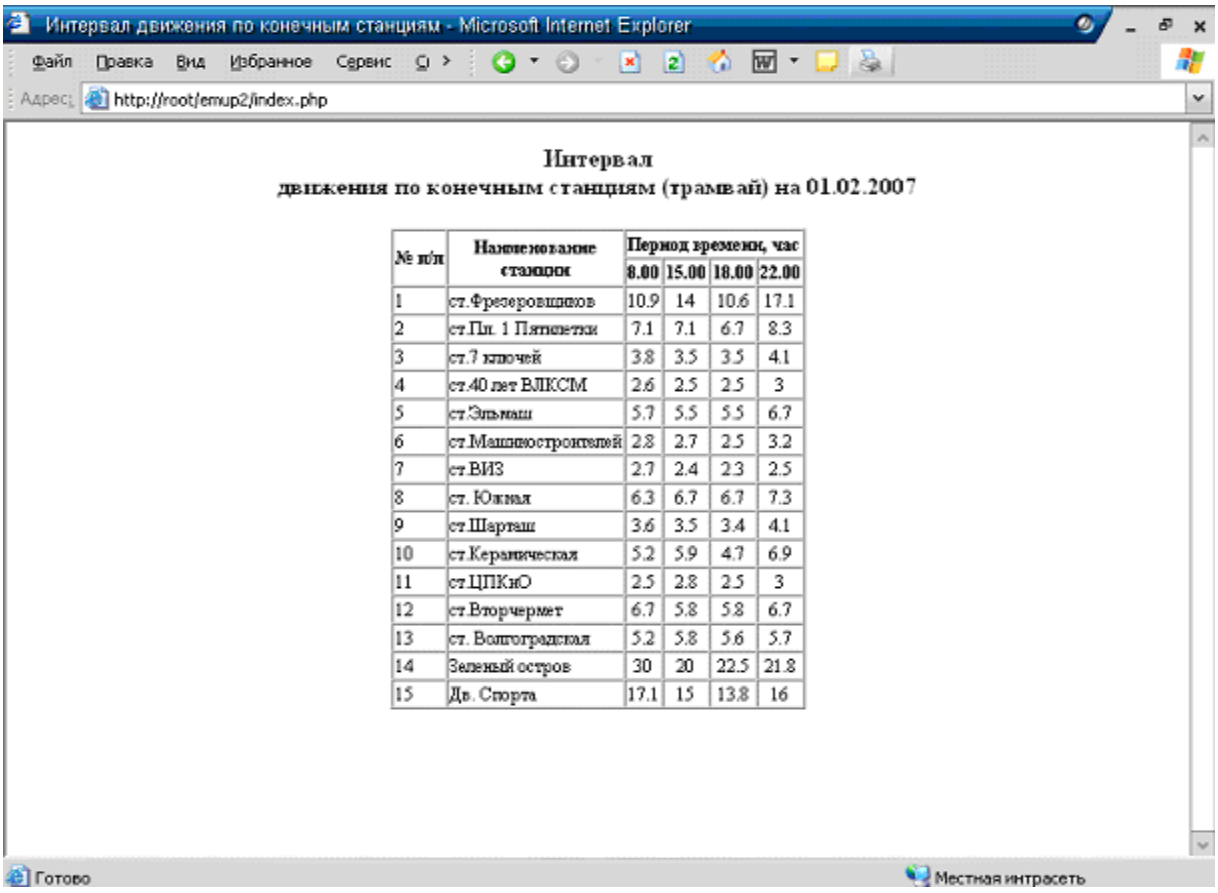


19.02.2007 09:16:12

Выпуск
подвижного состава ЕМУП «ТТУ»
по трамвайным маршрутам
с 1 февраля 2007 по 10 февраля 2007

М-т	8.00			15.00			18.00			22.30		
	план	факт	откл	план	факт	откл	план	факт	откл	план	факт	откл
1	101/101	101/101		87/ 87	87/ 87		101/101	101/101		10/ 10	10/ 10	
2	60/ 14	59/ 15	-1/ 1	32/	31/ 1	-1/ 1	42/ 14	42/ 14				
3	74/	74/		67/	66/	-1/	71/	71/		20/	20/	
4	99/ 18	99/ 18		82/ 16	83/ 14	1/ -2	97/ 16	95/ 17	-2/ 1	27/	27/ 4	/ 4
5	93/ 7	91/ 7	-2/	80/ 7	79/ 7	-1/	94/ 7	93/ 7	-1/	20/	20/	
6	101/ 44	101/ 44		87/ 37	87/ 39	/ 2	101/ 44	101/ 44		17/ 7	17/ 5	/ -2
7	87/ 37	86/ 40	-1/ 3	80/ 37	79/ 37	-1/	90/ 40	88/ 39	-2/ -1	20/ 10	21/ 9	1/ -1
8	106/106	105/105	-1/ -1	87/ 87	86/ 86	-1/ -1	107/107	105/104	-2/ -3	20/ 20	21/ 20	1/
9	42/	42/		14/	14/		42/	42/				
10	100/ 30	100/ 30		86/ 30	86/ 25	/ -5	100/ 30	99/ 30	-1/	30/	30/ 8	/ 8
11	20/	20/		20/	20/		20/	20/		10/	10/	
13	70/ 70	70/ 70		63/ 63	63/ 63		70/ 70	70/ 69	/ -1	10/ 10	10/ 10	
13(*)	60/ 60	60/ 60		53/ 53	52/ 52	-1/ -1	60/ 60	59/ 59	-1/ -1	10/ 10	11/ 11	1/ 1
14	87/ 55	87/ 55		73/ 41	73/ 52	/ 11	87/ 55	87/ 61	/ 6	30/ 27	31/ 20	1/ -7
15	157/132	157/132		133/108	133/112	/ 4	157/132	157/131	/ -1	53/ 42	53/ 40	/ -2
16	40/	37/	-3/	30/	30/		37/	36/	-1/	20/	20/	

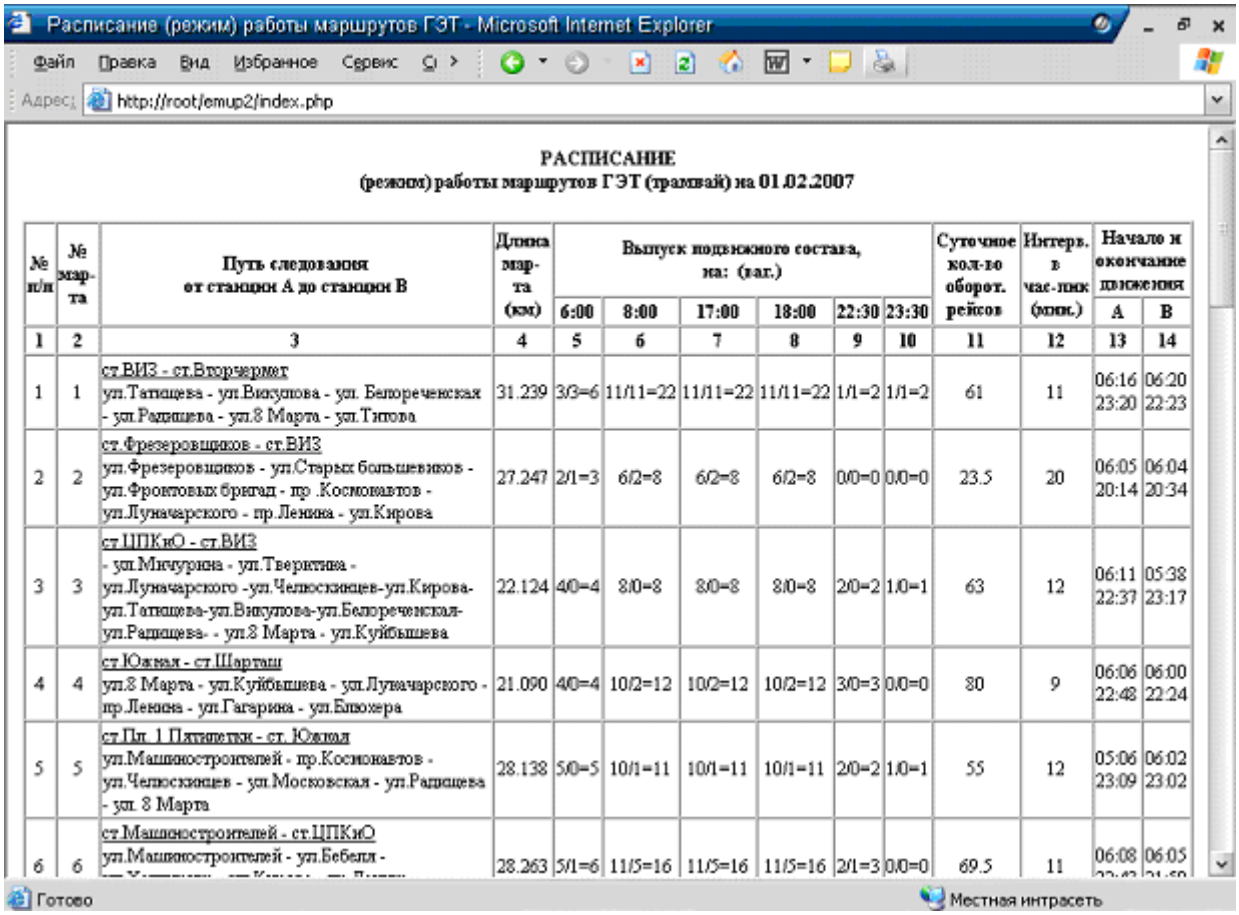
Рис. 4.37. Выпуск подвижного состава трамваев и троллейбусов по времени



Интервал
движения по конечным станциям (трамвай) на 01.02.2007

№ п/п	Наименование станции	Период времени, час			
		8.00	15.00	18.00	22.00
1	ст.Фрезеровщиков	10.9	14	10.6	17.1
2	ст.Пл. 1 Пятилетки	7.1	7.1	6.7	8.3
3	ст.7 ключей	3.8	3.5	3.5	4.1
4	ст.40 лет ВЛКСМ	2.6	2.5	2.5	3
5	ст.Эльмаш	5.7	5.5	5.5	6.7
6	ст.Машиностроителей	2.8	2.7	2.5	3.2
7	ст.ВИЗ	2.7	2.4	2.3	2.5
8	ст. Южная	6.3	6.7	6.7	7.3
9	ст.Шарташ	3.6	3.5	3.4	4.1
10	ст.Керамическая	5.2	5.9	4.7	6.9
11	ст.ЦПКиО	2.5	2.8	2.5	3
12	ст.Вторчермет	6.7	5.8	5.8	6.7
13	ст. Волгоградская	5.2	5.8	5.6	5.7
14	Зеленый остров	30	20	22.5	21.8
15	Дв. Спорта	17.1	15	13.8	16

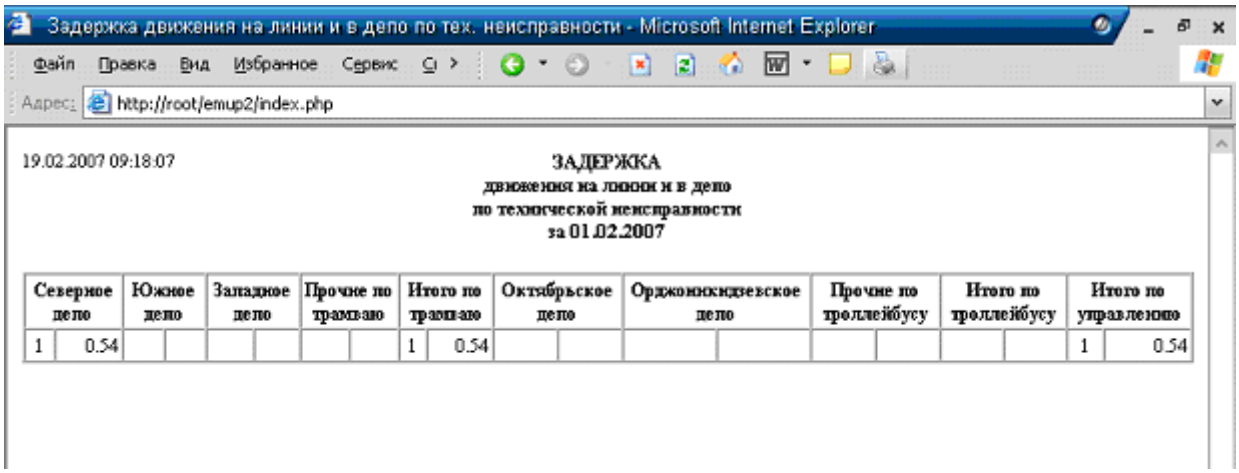
Рис. 4.38. Интервал движения по маршрутам по периодам времени трамваев



РАСПИСАНИЕ
(режим) работы маршрутов ГЭТ (трамвай) на 01.02.2007

№ п/п	№ мар-та	Путь следования от станции А до станции В	Длина мар-та (км)	Выпуск подвижного состава, на: (кат.)						Суточное кол-во оборот. рейсов	Интерв. в час-пик (мин.)	Начало и окончание движения	
				6:00	8:00	17:00	18:00	22:30	23:30			A	B
1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	ст.ВИЗ - ст.Вторчермет ул.Татищева - ул.Викулова - ул. Белореченская - ул.Радищева - ул.8 Марта - ул.Титова	31.239	3/3=6	11/11=22	11/11=22	11/11=22	1/1=2	1/1=2	61	11	06:16 23:20	06:20 22:23
2	2	ст.Фрезеровщиков - ст.ВИЗ ул.Фрезеровщиков - ул.Старых большевиков - ул.Фронтовых бригад - пр .Космонавтов - ул.Луначарского - пр.Ленина - ул.Кирова	27.247	2/1=3	6/2=8	6/2=8	6/2=8	0/0=0	0/0=0	23.5	20	06:05 20:14	06:04 20:34
3	3	ст.ЦПКиО - ст.ВИЗ - ул.Мичурина - ул.Тверитина - ул.Луначарского -ул.Челюскинцев-ул.Кирова-ул.Татищева-ул.Викулова-ул.Белореченская-ул.Радищева- - ул.8 Марта - ул.Куйбышева	22.124	4/0=4	8/0=8	8/0=8	8/0=8	2/0=2	1/0=1	63	12	06:11 22:37	05:38 23:17
4	4	ст.Южная - ст.Шарташ ул.8 Марта - ул.Куйбышева - ул.Луначарского - пр.Ленина - ул.Гагарина - ул.Блюхера	21.090	4/0=4	10/2=12	10/2=12	10/2=12	3/0=3	0/0=0	80	9	06:06 22:48	06:00 22:24
5	5	ст.Пл. 1 Пятилетки - ст. Южная ул.Машиностроителей - пр.Космонавтов - ул.Челюскинцев - ул.Московская - ул.Радищева - ул. 8 Марта	28.138	5/0=5	10/1=11	10/1=11	10/1=11	2/0=2	1/0=1	55	12	05:06 23:09	06:02 23:02
6	6	ст.Машиностроителей - ст.ЦПКиО ул.Машиностроителей - ул.Бебеля -	28.263	5/1=6	11/5=16	11/5=16	11/5=16	2/1=3	0/0=0	69.5	11	06:08	06:05

Рис. 4.39. Расписание (режим) работы трамваев и троллейбусов



Задержка движения на линии и в депо по тех. неисправности - Microsoft Internet Explorer

Файл Правка Вид Избранное Сервис

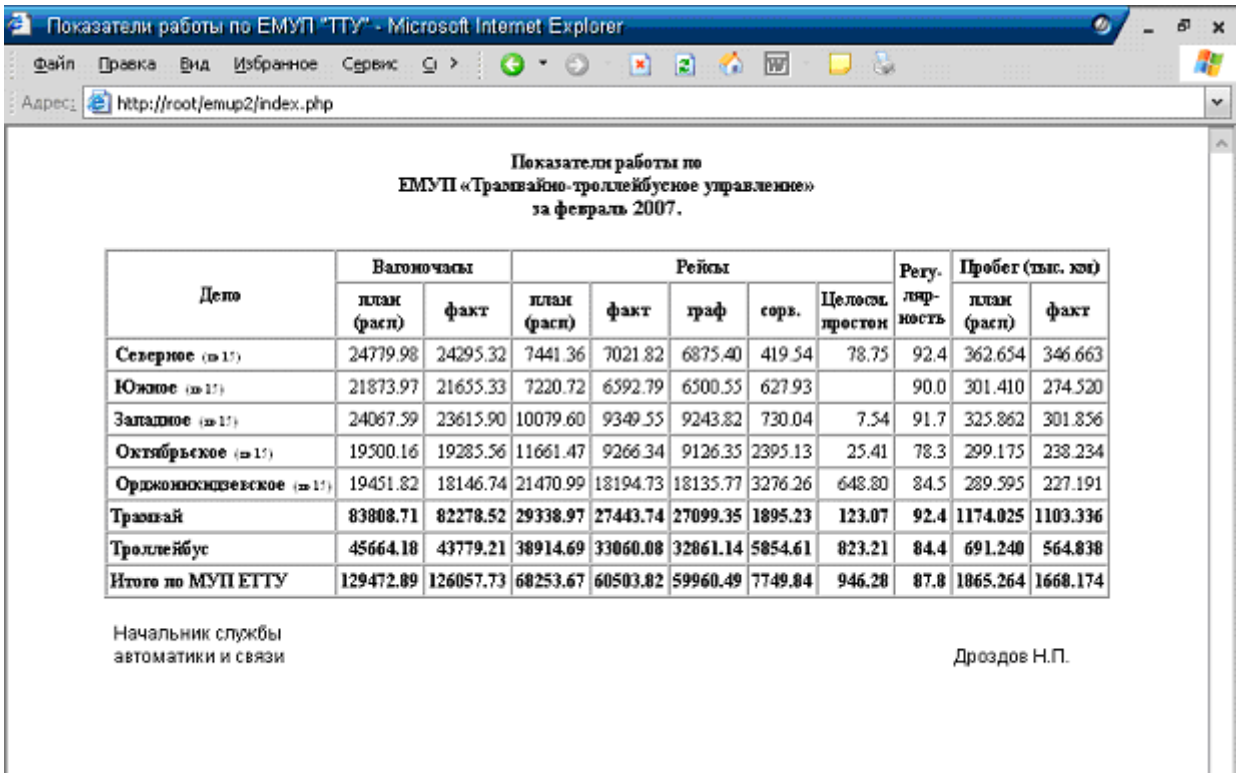
Адрес: http://root/emup2/index.php

19.02.2007 09:18:07

ЗАДЕРЖКА
движения на линии и в депо
по технической неисправности
за 01.02.2007

Северное депо		Южное депо		Западное депо		Прочие по трамваю		Итого по трамваю		Октябрьское депо		Орджоникидзевское депо		Прочие по троллейбусу		Итого по троллейбусу		Итого по управлению	
1	0.54							1	0.54									1	0.54

Рис. 4.40. Задержки движения на линии и в депо по техническим неисправностям



Показатели работы по ЕМУП "ТТУ" - Microsoft Internet Explorer

Файл Правка Вид Избранное Сервис

Адрес: http://root/emup2/index.php

Показатели работы по
ЕМУП «Трамвайно-троллейбусное управление»
за февраль 2007.

Депо	Вагоночасы		Рейсы					Регулярность	Пробег (тыс. км)	
	план (расп)	факт	план (расп)	факт	граф	сорв.	Целосм. простои		план (расп)	факт
Северное (в т.ч.)	24779.98	24295.32	7441.36	7021.82	6875.40	419.54	78.75	92.4	362.654	346.663
Южное (в т.ч.)	21873.97	21655.33	7220.72	6592.79	6500.55	627.93		90.0	301.410	274.520
Западное (в т.ч.)	24067.59	23615.90	10079.60	9349.55	9243.82	730.04	7.54	91.7	325.862	301.856
Октябрьское (в т.ч.)	19500.16	19285.56	11661.47	9266.34	9126.35	2395.13	25.41	78.3	299.175	238.234
Орджоникидзевское (в т.ч.)	19451.82	18146.74	21470.99	18194.73	18135.77	3276.26	648.80	84.5	289.595	227.191
Трамвай	**83808.71**	**82278.52**	**29338.97**	**27443.74**	**27099.35**	**1895.23**	**123.07**	**92.4**	**1174.025**	**1103.336**
Троллейбус	**45664.18**	**43779.21**	**38914.69**	**33060.08**	**32861.14**	**5854.61**	**823.21**	**84.4**	**691.240**	**564.838**
Итого по МУП ЕТТУ	**129472.89**	**126057.73**	**68253.67**	**60503.82**	**59960.49**	**7749.84**	**946.28**	**87.8**	**1865.264**	**1668.174**

Начальник службы
автоматики и связи

Дроздов Н.П.

Рис. 4.41. Показатели работы МУП ЕТТУ

На основе данных информационного сайта «Транспорт города Екатеринбурга» Комитет по транспорту и организации дорожного движения может:

- осуществлять ежедневный контроль выполнения основных показателей ЕМУП ТТУ;
- получать итоговые данные выполнения показателей работы ЕМУП ТТУ за период (за любой день, за любой месяц, за любой год);
- получать ежедневный контроль за регулярностью движения по маршрутам трамваев и троллейбусов, а также итоговые данные за прошлый день, за прошлый месяц;

- знать о выпуске подвижного состава на улицы города в любой заданный момент.

Все данные расписания выдаются за конкретно выбранный день, учитывая рабочие, выходные и праздничные дни, как календарные, так и дни переносов рабочих и выходных дней. Оперативность данных позволяет наиболее полно информировать население об изменениях в движении трамваев и троллейбусов, как постоянных (например, ввод нового маршрута или открытие новых участков движения), так и временных (во время закрытий определенных участков движения для производства ремонтных работ).

4.8. КОНТРОЛЬНЫЕ ВОПРОСЫ И УПРАЖНЕНИЯ

1. Из каких модулей состоит информационно-коммуникационной системы МУП ТТУ г. Екатеринбурга?
2. Посетите сайт МУП ТТУ г. Екатеринбурга *www.ettu.ru*. Работа каких модулей представлена в публичной части сайта?
3. В какие модули целесообразно включить вычисление критериев качества обслуживания транспортной сети электротранспорта и графика его работы (например, см. п. 3.13)?
4. За какие критерии качества обслуживания отвечает ЕМУП ТТУ, а за какие – администрация г. Екатеринбурга?
5. Определите, в функцию какого модуля входит обработка информации о задержках транспортных маршрутах и их причинах.
6. Постройте альтернативную схему базы данных для хранения исходных данных программного комплекса «Путевой лист автотранспортной службы» (см. рис. 4.34).
7. Создайте *SQL*-запросы для оценки рейсов по следующим показателям: пробег автомашины, фактический расчет горючего, экономия или перерасход горючего, ночное время работы водителя.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК К ГЛАВЕ 4

4.1. Дружинина Н.Г. Информационно-коммутационная система МУП ТТУ г. Екатеринбурга / Н.Г. Дружинина, О.Г. Трофимова. Научные труды международной научно-практической конференции «Связь-Пром 2008» в рамках V Евро-Азиатского международного форума «Связь-Ппромэкспо 2008». Екатеринбург: ЗАО «Компания Реал-Медиа», 2008. С. 226 – 228.

4.2. Дружинина Н.Г. Программный комплекс по составлению расписания маршрутизированного транспорта / Н.Г. Дружинина. Свидетельство об отраслевой регистрации разработки № 5198. Зарегистрировано в Отраслевом фонде алгоритмов и программ. 21.09.2005.

4.3. Дружинина Н.Г. Программный комплекс по составлению расписания маршрутизированного транспорта / Н.Г. Дружинина. М: ВНТИЦ, 2005.

№ 50200501360. Государственная регистрация в «Национальном информационном фонде неопубликованных документов» разработки, предъявленной в Отраслевой фонд алгоритмов и программ. 28.09.2005.

4.4. Дружинина Н.Г. Программный комплекс по обработке путевого листа автотранспортной службы / Н.Г. Дружинина. Свидетельство об отраслевой регистрации разработки № 5546. Зарегистрировано в Отраслевом фонде алгоритмов и программ. 29.12.2005.

4.5. Дружинина Н.Г. Программный комплекс по обработке путевого листа автотранспортной службы / Н.Г. Дружинина. М: ВНТИЦ, 2006. № 50200600033. Государственная регистрация в «Национальном информационном фонде неопубликованных документов» разработки, предъявленной в Отраслевой фонд алгоритмов и программ. 24.01.2006.

4.6. Дружинина Н.Г. Диспетчер выпуска и движение подвижной единицы / Н.Г. Дружинина. Свидетельство об отраслевой регистрации разработки № 6785. Зарегистрировано в Отраслевом фонде алгоритмов и программ. 22.08.2006.

4.7. Дружинина Н.Г. Диспетчер выпуска и движение подвижной единицы / Н.Г. Дружинина. М: ВНТИЦ, 2006. № 5020061544. Государственная регистрация в «Национальном информационном фонде неопубликованных документов» разработки, предъявленной в Отраслевой фонд алгоритмов и программ. 28.08.2006.

4.8. Дружинина Н.Г. Транспорт города Екатеринбурга / Н.Г. Дружинина, О.Г. Трофимова. Свидетельство об отраслевой регистрации разработки № 9916. Зарегистрировано в Отраслевом фонде алгоритмов и программ. 29.01.2008.

4.9. Дружинина Н.Г. Транспорт города Екатеринбурга / Н.Г. Дружинина, О.Г. Трофимова. М.: ВНТИЦ, 2008. № 50200800285. Государственная регистрация в «Национальном информационном фонде неопубликованных документов» разработки, предъявленной в отраслевой фонд алгоритмов и программ. 12.02.2008.

ЗАКЛЮЧЕНИЕ

Отметим основные перспективы использования данного пособия в научном и прикладном планах, а также в учебном процессе.

Развитие системы оценки качества сети городского транспорта и расписания маршрутизированного транспорта позволит усовершенствовать систему контроля администрации за работой ЕМУП ТТУ, учитывать потребности населения в быстром перемещении в большом мегаполисе. Сопровождение публичного графика работы маршрутизированного транспорта и предоставление пассажиру возможности выбора траектории перемещения существенно повышает коммуникативный обмен между администрацией города и населением.

Министерство транспорта России в 2009 г. создало департамент, занимающийся разработкой государственной политики в области дорожного хозяйства, в том числе организацией дорожного движения в городах. Эта организация станет ответственной за борьбу с «пробками», и предложенные в данном пособии подходы могут помочь ее деятельности при решении транспортных проблем. Пособие нацелено на развитие информационно-коммуникационной системы г. Екатеринбурга до современного уровня зарубежных городов. Рассмотренный набор моделей систем управления городским транспортом, дополненный интеллектуальной системой управления перекрестками, позволит реформировать сложную систему транспортных потоков мегаполиса.

Студенты получили уникальную возможность рассмотреть реализацию оригинальной информационно-коммуникационной системы крупного предприятия ЕМУП ТТУ. Это приведет к более глубокому усвоению и усовершенствованию знаний в области универсальных инструментальных средств проектирования систем, поможет в разработке собственных сложных информационных систем.

С другой стороны, содержание второй главы, посвященной специализированным инструментальным средствам имитационного моделирования систем управления, позволит студентам практически освоить и закрепить навыки имитационного моделирования систем управления различных типов, например систем массового обслуживания, динамических систем, системы автоматического регулирования. Некоторые аспекты транспортных задач являются частными случаями систем управления. Таким образом, рассмотренные в третьей и четвертой главах транспортные системы могут исследоваться и развиваться с использованием инструментальных средств имитационного моделирования, особенно специальных средств обработки и анализа результатов стохастического моделирования. Например, для систем массового обслуживания обязательно анализируется загрузка элементов системы, максимальная и средняя длина очереди, среднее время между поступлениями новых заявок, необходимое количество каналов для

разгрузки системы. Аналогичные характеристики рассматриваются и в транспортной задаче для определения качества работы. После создания сложной интеллектуальной системы управления перекрестками анализ ее работы можно провести также с помощью имитационного моделирования. При этом для управления совокупностью смежных городских перекрестков можно использовать систему автоматического регулирования, применяя стохастические методы для анализа состояния транспортных потоков.

Учебное издание

Лисиенко Владимир Георгиевич
Дружинина Надежда Геннадьевна
Трофимова Ольга Геннадиевна
Трофимов Сергей Павлович

МОДЕЛИРОВАНИЕ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Редакторы *Н.П. Кубыщенко, О.В.Протасова*
Компьютерный набор *Н.Г. Дружининой, О.Г. Трофимовой*

Подписано в печать	30.07.2009	Формат 60х84 1/16
Бумага писчая	Плоская печать	Усл. печ. л. 25,58
Уч.-изд. л. 21,2	Тираж 100 экз.	Заказ

Редакционно-издательский отдел УГТУ – УПИ
620002, Екатеринбург, ул. Мира, 19
rio@mail.ustu.ru

Ризография НИЧ УГТУ – УПИ
620002, Екатеринбург, ул. Мира, 19